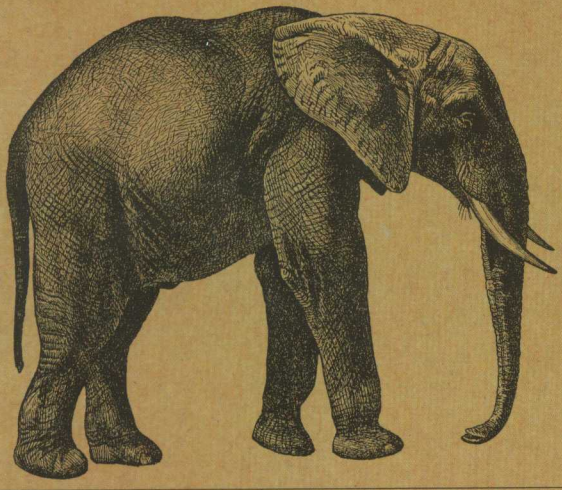
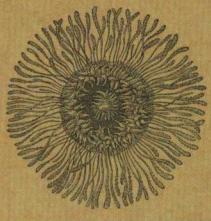
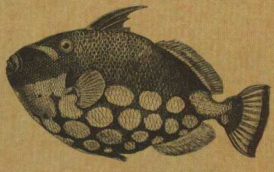
O'REILLY®



Hadoop
生态系统

Field Guide to Hadoop


KEVIN SITTO & MARSHALL PRESSER 著
陈新 唐晓 译

中国电力出版社

Hadoop生态系统

Kevin Sitto & Marshall Presser 著

陈新 唐晓 译

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo O'REILLY®

O'Reilly Media, Inc.授权中国电力出版社出版

中国电力出版社

图书在版编目（CIP）数据

Hadoop生态系统/（美）凯文•斯托（Kevin Sitto），（美）马歇尔•普瑞斯（Marshall Presser）著；陈新，唐晓译. —北京：中国电力出版社，2016.11

书名原文：Field Guide to Hadoop

ISBN 978-7-5123-9598-5

I. ①H… II. ①凯… ②马… ③陈… ④唐… III. ①数据处理软件 IV. ①TP274

中国版本图书馆CIP数据核字（2016）第174896号

北京市版权局著作权合同登记

图字：01-2016-4203号

封面设计/ Ellie Volckhausen，张健

出版发行/ 中国电力出版社（http://www.cepp.sgcc.com.cn）

地　　址/ 北京市东城区北京站西街19号（邮政编码100005）

经　　销/ 全国新华书店

印　　刷/ 北京天宇星印刷厂

开　　本/ 787毫米×980毫米　16开本　7.75印张　123千字

版　　次/ 2016年11月第一版　2016年11月第一次印刷

印　　数/ 0001－3000册

定　　价/ 28.00元（册）

O'Reilly Media, Inc.介绍

O'Reilly Media通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自1978年开始，O'Reilly一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了Make杂志，从而成为DIY革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O'Reilly的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版，在线服务或者面授课程，每一项O'Reilly的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar博客有口皆碑。”

——Wired

“O'Reilly凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——Business 2.0

“O'Reilly Conference是聚集关键思想领袖的绝对典范。”

——CRN

“一本O'Reilly的书就代表一个有用、有前途、需要学习的主题。”

——Irish Times

“Tim是位特立独行的商人，他不光放眼于最长远、最广阔的视野并且切实地按照Yogi Berra的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去Tim似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——Linux Journal

感谢我漂亮的妻子Erin，感谢她无限的耐心，也要送给我优秀的孩子们Dominic和Ivy，他们把我照顾得很好。

——Kevin

感谢我的妻子Nancy Sherman，感谢她在我写作、改写和再次改写的过程中不断的鼓励。同样，要非常感谢可爱的小黄象，若没有它，我们也不会想到写这样一本书。

——Marshall

目录

前言

Hadoop是什么？为什么需要关注它？本书将帮助你理解Hadoop是什么，现在先让我们来回答一下第二个问题。Hadoop是当前最流行的可以支持存储和大数据分析的独立平台。如果你和你的组织计划进入大数据这个激动人心的世界，那么你将必须决定Apache Hadoop是否是一个适合使用的平台，同时还需要决定Hadoop中哪些组件最适合你的任务。本书的作用就是为你介绍Hadoop的各个主题，开启你的Hadoop旅程。

现在已经有许多关于Hadoop及其相关技术的书籍、网站和培训班。本书与众不同的地方是并没有提供一个关于Hadoop特定方面或Hadoop生态系统中关于各种组件的冗长的教程介绍。本书并不是一本详细讨论Hadoop中任何一个主题的书籍。相反，与鸟语林中的路标一样， 每一章都聚焦一个拥有共同主题的Hadoop生态系统中的一部分。在每一章中，简要介绍了相关的技术和主题：我们解释了该技术与Hadoop的关系，并讨论了为什么该技术适用某些需求（以及在某些场景下并不适用）。最后，本书包含了多个小节内容，这些小节讨论了多个Apache Hadoop的项目和子项目及其相关技术，并给出了相关技术和方法的教程链接。

在每一小节，通常包含类似下面的一个表格：

许可证	< 许可证信息 >
活跃度	< 无，低，中，高 >
用途	< 相关用途 >

官方网站　　< 网址 >

Hadoop 集成度　　完全集成、API 兼容、未集成、不适配

现在详细看看这些类别的描述：

许可证（License）

尽管本书中各个小节中的软件都是开源的，但这些软件使用了不同的许可证，大体上是一样的，但也有一些小的不同。如果你有计划在产品中使用这些软件，那么你就需要对这些许可证的使用条件比较熟悉。

活跃度（Activity）

我们尽可能地度量了该技术在开发中使用的活跃度。可能在某些场景下我们的度量是有错误的，并且活跃度等级可能在撰写完本主题后也会发生改变。

用途（Purpose）

该技术可以用来做什么？我们将尽力根据用途的相似度将主题进行分组，但是有时候我们发现某一个主题可以适用不同章节。生活就是不断做出选择，这就是我们的选择。

官网

对于本书涉及的技术，互联网上最权威的站点可能就是它的官方网站了。

Hadoop 集成度

在写作时，我们并不确定有哪些主题会包含在本书的第一个版本中。最原始清单中的一些主题是紧密集成或者说绑定在 Hadoop 中的。有一些其他技术是可选择的，或者是可以和 Hadoop 共同工作的，但并不是 Apache Hadoop 家庭中的一部分。对于那些情形，我们尽可能描述在写作的时候该技术的集成度。当然随着时间变化，毫无疑问这将会发生改变。

本书并不需要从头到尾阅读。如果你完全是一个新的 Hadoop 用户，那么你可以从第 1 章（介绍）开始阅读。然后可以看看感兴趣的主题，阅读关于该组件的小节内容，阅读章节的章下文，也可以浏览同一章中的其他小节。本书通常包含了指向其他可能相关小节的链接。你也可以查看关于该主题的教程或者官方网站。

我们按照图 P-1 所示的模式，将所有主题分类为多个章节。许多主题纳入 Hadoop Common（之前称为 Hadoop Core），这是支持所有其他 Apache Hadoop 模块的最基础的工具和技术。然而，这些工具在大数据生态系统中也起到了很重要的作用，并不仅仅局限于是 Hadoop 的内核技术。在本书中我们也讨论了许多在大数据视角起了重要作用的相关技术。

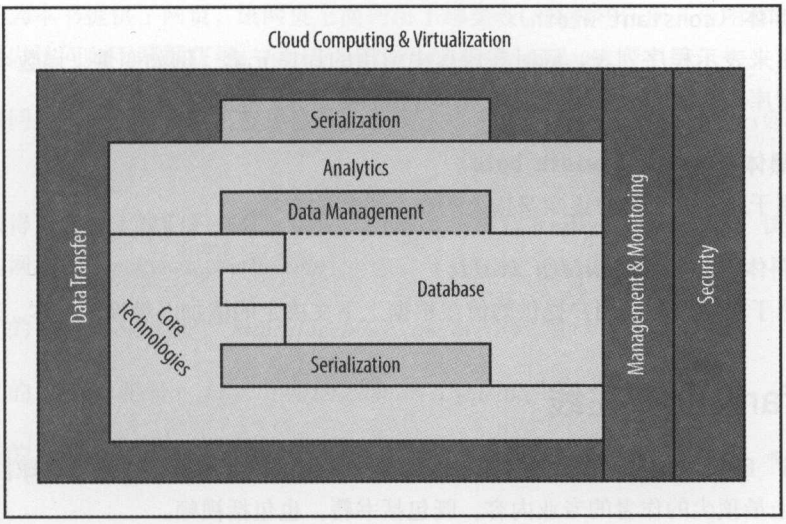


图 P-1：本书涉及主题的概览

在本书中，我们并没有包含任何专用的 Hadoop 发行版本。我们认识到这些项目是重要并且有重大意义的，但是商业系统变化特别快，所以我们重点只关注开源技术。

当前，开源技术在 Hadoop 和大数据市场中有很重要的影响力，许多商业的解决方案都是深度基于本书中描述的开源技术来实现的。如果读者对本书中讨论的开源技术感兴趣，有采用该技术的倾向，也可以去寻找商业发行版本。

本书并不是一个每年或每两年才更新的一个静态文档。我们的目标是尽可能地保持内容最新，随着 Hadoop 环境的增长而增加新的内容，有一些陈旧的技术要么会消失，要么进入维护模式，因为它们可能会被其他能够满足更新技术需求或由于其他原因获得支持的技术所代替。

因为本书涉及的主题可能会变化很快，非常欢迎读者将建议和评论反馈给 Kevin(ksitto@gmail.com) 和 Marshall(bigmaish@gmail.com)。你们有任何建议都可以反馈给我们，我们将十分感谢。

本书使用的排版约定

本书使用如下排版约定：

斜体 (italic)

用来表示新术语、URL、email 地址、文件名、文件扩展名等。

等宽字体（`constant width`）

用来表示程序列表，同时在段落中引用的程序元素（例如变量、函数名、数据库、数据类型、环境变量、声明和关键字等）也用该格式表示。

等宽黑体（`constant width bold`）

用于表示需要用户逐字符输入的命令或其他文本。

等宽斜体（*`constant width italic`*）

用于表示应该以用户提供的值或根据上下文决定的值加以替换的文本。

Safari® 图书在线

Safari® Book Online 是一个按需定制的数字化图书馆，它提供了来自全球技术和业务上最顶尖的作者的专业内容，既包括书籍，也包括视频。

技术专业人员、软件开发者、网页设计者和商业创新专业人员都可以使用 Safari Book Online 来作为研究问题、解决、学习和认证培训的主要资源。

Safari Book Online 为组织者、政府机构和个人提供了各种价格范围的资源。订阅者可以通过一个统一的可搜索数据库访问成千上万的书籍、培训视频和预先出版的手稿，提供资源的出版社包括 O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology 等。如需更多关于 Safari Book Online 的信息，请访问我们的网站。

如何联系我们

美国：

O'Reilly Media，Inc.

1005 Gravenstein Highway North

Sebastopol，CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）

奥莱利技术咨询（北京）有限公司

我们为本书提供了网页，该网页上面列出了勘误表、范例和任何其他附加的信息。您可以访问如下网页获得：*http:// bit.ly/field-guide-hadoop*。

要询问技术问题或对本书提出建议，请发送电子邮件至：*bookquestions@oreilly.com*

要获得更多关于我们的书籍、会议、资源中心和 O'Reilly 网络的信息，请参见我们的网站：*http://www.oreilly.com*

我们的 Facebook 地址：*http://facebook.com/oreilly*

我们的 Twitter 地址：*http://twitter.com/oreillymedia*

我们的 YouTube，地址：*http://www.youtube.com/oreillymedia*

致谢

我们要感谢本书的评审 Harry Dolan、Michael Park、Don Miner 和 Q Ethan McCallum。非常感谢他们为此付出的时间，以及洞察力和耐心。

我们还要非常感谢 O'Reilly 团队给我们的帮助。我们特别希望感谢 Mike Loukides，感谢他在我们刚开始写作时给我们提供的宝贵的帮助，还要感谢 Ann Spencer，感谢她帮助我们理清如何写作本书的思路，要感谢 Shannon Cutt，他们提供的评论使得本书成为可能。另外要特别感谢 Rebecca Demarest 和 Dan Fauxsmith。

此外，还要特别感谢 Paul Green，感谢他在大数据还是一个不为人知的小事物时，教会我们什么是大数据，还要感谢 Don Brancato，感谢他强迫一个程序员来阅读 strunk and white 写的《the elements of style》[译注 1]。

译注 1: 这是一本英语语法书籍。

第 1 章

关键技术

在 2002 年，当时万维网（World Wide Web）相对来说还比较新，大家也还没有使用谷歌来搜索东西，Doug Cutting 和 Mike Cafarella 想要抓取网页并对其内容做索引，以此来开发一个因特网的搜索引擎。他们启动了一个名为 Nutch 的项目来做这件事情，该项目需要通过一个可伸缩的方法来存储这些索引的内容。在 2002 年，通常采用关系数据库系统（RDBMS）来组织和存储数据的标准方法，该系统用一种名为 SQL 的语言来存取数据。但是几乎所有的 SQL 和关系存储都不适合因特网上搜索引擎的存储和检索。它们很昂贵，不具备可伸缩性，不能接受失败请求，并且可能达不到所需的性能。

在 2003 年和 2004 年，谷歌发表了两篇重要的论文，一篇是有关 Google 文件系统的（*http://bit.ly/1CgWGTy*），另一篇是关于集群服务器上名为 MapReduce（*http://bit.ly/12c3Ifq*）的编程模型的。Cutting 和 Cafarella 将这些技术应用到他们的项目中，最终 Hadoop 就产生了。Hadoop 不是一个首字母的缩写。Cutting 的儿子有一个黄色的毛绒大象，它的名字叫 Hadoop，然后不知怎么，这个名字就被标记在了这个项目上，并且这只伶俐可爱的小象就成为它的图标。Yahoo! 最开始将 Hadoop 作为它搜索引擎的基础，然后很快就扩展到了很多其他的组织。现在 Hadoop 是最主要的大数据平台。已有很多详细描述 Hadoop 的资源。在这里将看到许多有关组件的概要描述，通过它们的指向还可以学习到更多的相关信息。

Hadoop 由三个主要的资源组成：

- Hadoop 分布式文件系统 (HDFS)。
- MapReduce 编程平台。
- Hadoop 生态系统，这是一个工具的集合，它们使用或在 MapReduce 和 HDFS 周围去存储和组织数据，以及对那些运行 Hadoop 的机器进行管理。

这些机器被称为一个集群，也就是一个服务器组，它们几乎总是运行在一些不同版本的 Linux 操作系统上去执行共同的任务。

Hadoop 生态系统由模块组成，这些模块帮助编写系统、管理及配置集群、管理集群中的数据和存储、执行分析任务，以及类似的工作。本书中的大部分模块将描述生态系统中的不同组件及相关的技术。

1.1 Hadoop 分布式文件系统（HDFS）

许可证	Apache License, Version 2.0
活跃度	高
用途	大容量、容错性、可存储非常大的数据集的廉价存储
官方网站	http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html
Hadoop 集成度	完全集成

Hadoop 分布式文件系统（HDFS）是 Hadoop 集群中的一部分，它是用来存储数据的。为了建造一个数据密集型应用，HDFS 的目的是在廉价的商业服务器上运行集群。HDFS 对高性能、阅读密集型操作进行了优化，且对集群上的错误有适应性。它不能阻止错误的发生，但是它也不太可能丢失数据，因为 HDFS 在默认情况下对它的每个数据块都存有多个副本。除此之外，HDFS 是一次写入、多次读取的文件系统：一旦一个文件被创建，文件系统 API 仅仅允许你去添加文件，而不允许你去重新改写它。因此，通常 HDFS 对在线事务处理系统（OLTP）不太适用。HDFS 最主要的用途是用于连续读取大型文件。这些文件将分解成大文件块，通常是 64MB 或者更大，这些块分布在服务器的各个节点之中。

HDFS 文件系统不兼容 POSIX，就好像你在 Linux、Mac OS X，以及一些 Windows 平台上看到的那样（可以查看 POSIX 维基百科（*http://bit.ly/16TI2GO*）获取相关的简要说明）。它不由服务器节点上的操作系统内核来管理。HDFS

上的块映射到主机底层文件系统的文件上，通常采用的是 Linux 系统的 ext3。HDFS 假定主机上底层的磁盘没有被 RAID 所保护，所以默认情况下，每个块都有三个备份，并且分布在不同的集群节点上。这就在节点或者磁盘损坏的时候为数据提供了保护，目的是为了防止数据丢失，并且有助于对 Hadoop 上的数据进行访问，而不是将这些数据通过网络移动后再访问它们。

尽管这些说明超过的本书的范围，HDFS 上有关文件的元数据是通过 NameNode 来管理的，Hadoop 相当于 UNIX/Linux 的超级块。

1.1.1 教程链接

通常情况下，你可以通过其他的一些工具与 HDFS 进行交互，如 Hive（详见第 2 章）或者 Pig（详见第 5 章）。即便如此，你想立刻用 HDFS 来工作还需要一些时间；Yahoo！已经出版了一本对这个基础系统进行配置和探索的非常优秀的指导说明（*http://yhoo.it/1uEUNQJ*）。

1.1.2 示例代码

当你使用一个 Hadoop 客户端的命令行界面（CLI）时，你可以将文件从本地复制到 HDFS 上，然后你可以通过下面的代码段查看前 10 行。

```
[hadoop@client-host ~]$ hadoop fs -ls /data
Found 4 items
drwxr-xr-x - hadoop supergroup 0 2012-07-12 08:55 /data/faa
-rw-r--r-- 1 hadoop supergroup 100 2012-08-02 13:29
/data/sample.txt
drwxr-xr-x - hadoop supergroup 0 2012-08-09 19:19 /data/wc
drwxr-xr-x - hadoop supergroup 0 2012-09-11 11:14 /data/weblogs

[hadoop@client-host ~]$ hadoop fs -ls /data/weblogs/

[hadoop@client-host ~]$ hadoop fs -mkdir /data/weblogs/in

[hadoop@client-host ~]$ hadoop fs -copyFromLocal
weblogs_Aug_2008.ORIG /data/weblogs/in

[hadoop@client-host ~]$ hadoop fs -ls /data/weblogs/in
Found 1 items
-rw-r--r-- 1 hadoop supergroup 9000 2012-09-11 11:15
/data/weblogs/in/weblogs_Aug_2008.ORIG

[hadoop@client-host ~]$ hadoop fs -cat
/data/weblogs/in/weblogs_Aug_2008.ORIG \
| head
10.254.0.51 - - [29/Aug/2008:12:29:13 -0700] "GGGG / HTTP/1.1"
200 1456
10.254.0.52 - - [29/Aug/2008:12:29:13 -0700] "GET / HTTP/1.1"
200 1456
```

```
10.254.0.53 - - [29/Aug/2008:12:29:13 -0700] "GET /apache_pb.gif
HTTP/1.1" 200 2326
10.254.0.54 - - [29/Aug/2008:12:29:13 -0700] "GET /favicon.ico
HTTP/1.1" 404 209
10.254.0.55 - - [29/Aug/2008:12:29:16 -0700] "GET /favicon.ico
HTTP/1.1"
404 209
10.254.0.56 - - [29/Aug/2008:12:29:21 -0700] "GET /mapreduce
HTTP/1.1" 301 236
10.254.0.57 - - [29/Aug/2008:12:29:21 -0700] "GET /develop/
HTTP/1.1" 200 2657
10.254.0.58 - - [29/Aug/2008:12:29:21 -0700] "GET
/develop/images/gradient.jpg
HTTP/1.1" 200 16624
10.254.0.59 - - [29/Aug/2008:12:29:27 -0700] "GET /manual/
HTTP/1.1" 200 7559
10.254.0.62 - - [29/Aug/2008:12:29:27 -0700] "GET
/manual/style/css/manual.css
HTTP/1.1" 200 18674
```

1.2 MapReduce

许可证	Apache License, Version 2.0
活跃度	高
用途	一种处理大数据的编程范式
官方网站	https://hadoop.apache.org
Hadoop 集成度	完全集成

MapReduce 是 Hadoop 上开发应用程序的第一个也是最主要的一个编程框架。你要使用最原始、最纯粹的 MapReduce 形式就需要用到 Java 语言。同时你也需要学习 WordCount 范例，也就是 Hadoop 中的“Hello, world”程序。所有的标准 Hadoop 发行版本中都包含该范例。下面是 WordCount 解决的问题：你有一个由非常多的文档组成的数据集，而目标是得到一个列表，该列表包括在该数据集中出现过的所有单词以及它们的出现频率。

MapReduce 的工作是由名为 mapper 和 reduce 的 Java 程序组成。对 Hadoop 软件精心策划，给每个 mapper 大量的数据来进行分析处理。让我们来假设它得到了这样一个句子。“The dog ate the food”它将得到 5 个名 - 值对（name-value pairs）或 map：“the”:1，“dog”:1，“ate”:1，“the”:1 和“food”:1。名 - 值对中的名称是单词，它的值是这些单词出现的次数。Hadoop 将获得 map 工作的结果并对该结果进行分类整理。并通过一个叫作 shuffle 的步骤为每一个 map 创建一个哈希值，并将其分配给 reducer。reducer 会在输入流中将对所有 map 中的每一个单词进行汇总，然后在文档中生成单词的分类列表。你可以将 mapper 看成是一个从 HDFS 文件中抽取数据放到 map 中的过程，将 reducer 看成是得到 mapper 中的输出然后汇总结果的过程。在下面的教程链接部分将对其有更为详细的解释。

你将很开心地知道这些复杂的工作都是靠 Hadoop 自己来完成的，分割输入的数据集，将各个 mapper 和 reducer 分配到节点上，将 mapper 上的数据打乱后放到 reducer 上，然后将最终的结果写到 HDFS 上。在程序中我们只需要写 map 和 reduce 函数。我们通常用 Java 来编写 mapper 和 reducer 程序（就好像在这部分结尾的地方引用的例子），但编写 MapReduce 代码对于新手来说是还是很复杂的。为了简化程序编写的复杂性，开发了更高层次的工具来完成这件事情。Pig 就是其中一个例子，我们将在第 5 章来讨论它。Hadoop Streaming 则是另外一个例子。

1.2.1 教程链接

网上有一些关于 MapReduce 作业的非常优秀的教程。Apache 官网文档（*http://bit.ly/1KMvKgv*）是一个非常好的开始，而 Yahoo! 同样也汇总了很好的教程模块（*http://yhoo.it/1MEF6ie*）。MapR，一个商业软件公司做的 Hadoop 发布，对于在 MapR 上编写 MapReduce 的人们来说这是一个非常好的选择（*http://youtu.be/kM76O4cZ5_0*）。

1.2.2 示例代码

编写 MapReduce 是相当复杂的，它远超过了本书涉猎的范围。人们常常从一个典型的应用程序——编写单词计数开始，官网上的文档中就有构建这个应用程序的教程（http://bit.ly/1yhaB7q）。

1.3 YARN

许可证	Apache License, Version 2.0
活跃度	中
用途	数据处理
官方网站	https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html
Hadoop 集成度	完全集成

一想到 Hadoop，人们真正想到的是两个相关的技术。这两个技术分别是 Hadoop 分布式文件系统（HDFS）和 MapReduce。HDFS 是装载数据的房子，MapReduce 则允许对数据做一些实际的操作。尽管 MapReduce 对某些任务来说特别有用，但是在其他方面也存在一些不足。如它导致了生态系统的分裂，在 Hadoop 集群外面有各种各样的工具，而它们又试图与 HDFS 进行通信。

在 2012 年 5 月，Hadoop 发布了 2.0 的版本，随着该版本的发布，一个激动人心的改变就是你可以与你的数据进行交互了。这个改变就是引入了 YARN，也就是 Yet Another Resource Negotiator。

YARN 位于你的数据之间，也就是现在 MapReduce 所在的位置，它允许过去那些在 Hadoop 系统外的其他的工具现在可存在于一个 Hadoop 集群中，如 Spark、Giragh。一个很重要的理解是，YARN 并不是取代了 MapReduce。实际上，YARN 本身不能做任何事情。YARN 为各类工具提供了一个便捷的、统一的方式，比如 MapReduce、HBase，或者是其他一些构建的可以在 Hadoop 集群上运行的自定义工具。

1.3.1 教程链接

YARN 仍然是一个进化的技术，Apache 的官网指南（*http://bit.ly/1E9z6ry*）是一个很不错的开始。

1.3.2 示例代码

实际情况是在 YARN 上编写应用程序仍然是一件非常复杂的事，且对本书来说也过于复杂。你可以在前面的“教程链接”部分找到非常优秀的详解链接去建立你的第一个 YARN 应用程序。

1.4 Spark

许可证	Apache License, Version 2.0
活跃度	高
用途	数据处理 / 存储
官方网站	http://spark.apache.org/
Hadoop 集成度	API 兼容

在大多数 Hadoop 的核心集群上 MapReduce 是最主要的部分。尽管 MapReduce 在大规模批处理分析工作上是非常高效的，但是对于那些类似需要迭代处理和数据共享的图形分析的应用程序来说，MapReduce 已被证实并不是最优的。

Spark 的设计提供了一个更为灵活的模型，这个模型支持许多在 MapReduce 上摇摇欲坠的多通道应用程序。为了达到这个目标，它尽可能地利用内存以减少从硬盘上写入或读取数据的数据总量。和 Pig 和 Hive 不一样，Spark 并不是让 MapReduce 成为更容易使用的一种工具。它是包括工作执行引擎在内的 MapReduce 的完全替代品。

Spark 操作的三个核心理念：

单行分布式数据集（RDD）

RDD 包含那些你想要转化或分析的数据。它们既可以从外部来源读取，比如文件或数据库，也可以通过一个 transformation 创建。

Transformation

一个 transformation 将一个现有的 RDD 修改后创建成一个新的 RDD。比如，一个 filter 将日志文件中的 ERROR 信息剔除就是一个 transformation。

Action

一个 action 分析一个 RDD 并且返回一个唯一的结果。比如，一个 action 可以通过识别我们的 ERROR filter 来计算结果的数量。

如果你想在 Spark 上做一些有意义的事情，学习 Scala 这种函数式编程语言是非

常明智的。Scala（http://www.scala-lang.org）将面向对象和函数程序设计相结合。因为 Lisp 是一种古老的函数式编程语言，Scala 也许可以称为“加入 21 世纪的 Lisp”。这并不是说 Scala 是在 Spark 上工作的唯一的方法。该项目同样对 Java 和 Pyhton 有强有力的支持，但是当增加新的 API 或产品特性时，它们是以 Scala 形式第一次出现的。

1.4.1 教程链接

在项目的主页上可以找到 Spark 的快速入门指南（ http://bit.ly/1upkhfx）。

1.4.2 示例代码

我们首先通过运行安装 Spark 目录上的 *./bin/spark-shell* 来打开 Spark shell。

在这个例子里，将计算评论文件中 Dune 所发表评论的数量。

```
// 读取包含评论的 csv 文件
scala> val reviews = spark.textFile("hdfs://reviews.csv")
testFile: spark.RDD[String] = spark.MappedRDD@3d7e837f

// 这是一个两部分的操作：
// 首先过滤两行包含有 Dune 的评论，
// 然后计算这些行
scala> val dune_reviews = reviews.filter(line =>
  line.contains("Dune")).count()
res0: Long = 2
```

第2章

数据库及数据管理

如果你计划使用 Hadoop，将很可能要对大量的数据进行管理，除了 MapReduce 的工作之外，你也许还需要各种各样的数据库。当 Google 的 BigTable 出现以后，Hadoop 对数据的管理变得越来越有兴趣。虽然现在已经有一些关系 SQL 数据库及 HDFS 数据的 SQL 接口，如 Hive，但是在 Hadoop 上对数据的管理大多使用的是 non-SQL 技术，通过 non-SQL 技术来对数据进行存储和访问。在 NoSQL Archive（*http://nosql-database.org/*）上列出了 150 多种 NoSQL 数据库，它们被分为以下几种类型：

- 列式存储。
- 文档存储。
- 键值 / 元组存储（Key-value/tuple stores）。
- 图数据库。
- 多模型数据库。
- 对象数据库。
- 网格和云数据库。
- 多值数据库。
- 表格存储。
- 其他。

NoSQL 数据库通常不支持关系连接操作、复杂事务，以及通常出现在关系系统中的外键约束，但是 NoSQL 数据库通常适合大规模的海量数据。

你需要决定哪种类型的数据库最适合你的数据集，以及你希望从中提取哪些信息，你很有可能用到不止一种类型的数据库。

本书将着眼于每一节中的主要案例，但是重点将放在两个类型上：键值存储和文档存储，如图 2-1 所示。

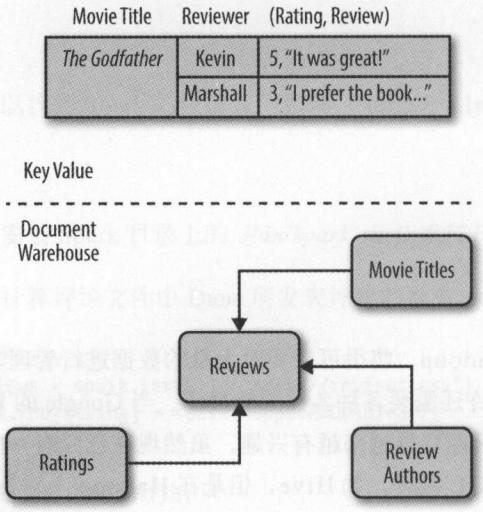


图 2-1：索引的两种方法

你可以将键值存储想象成一个目录。目录中所有的项（值）围绕着某种类型的索引（键）。就好像在一个目录中，如果你知道你要查找的关键字，那么使用键值存储是非常快速高效的，但是如果你不知道你要查找的关键字，那么它就没办法给你能提供更多的帮助了。

比如说，我们想查找 Marshall 对 Godfather 的评论。我们可以快速的查阅我的索引，找到所有这部电影的评论，然后向下滚动找出 Marshall's 的评论："我喜欢这本书"。

另一方面，文档存储是一种更为灵活的数据库类型。它并不会强迫你围绕一个指定键来组织数据，相反的，它允许你基于任意数量的参数来索引和搜索数据。让我们对刚才最后一个例子进行详细描述，如果说，我因为一本书想要去看一部电影。那么要通过搜索包含单词"书"的评论找到这样一部电影，这是多么幼稚的方法啊。

在这个例子中，我要使用的关键字不是很明确，这时键值存储将不会给我们提供很多的帮助。我们需要的是文档存储，它可以让我们快速地查找所有评论中的文本，然后找到那些包含单词"书"的内容。

2.1 Cassandra

许可证	GPL v2
活跃度	高
用途	键值存储
官方网站	https://cassandra.apache.org
Hadoop 集成度	API 兼容

通常情况下，为了能实现方便的检索，你可能需要对大数据进行简单的组织。要完成这个工作的一个常用的方法就是使用键值（key-value）存储。这种类型的数据库看起来就像是电话本上的白纸。数据可用一个统一的“key”来组织，而其value 就是与这个 key 相关联的信息。比如，如果要存储客户的信息，可以将他们的名字作为 key，而将那些与这个 key 相关联的信息比如往来业务、地址等作为 value。

键值存储在大数据系统上是一种常用的方式，因为它易于扩展，快速且工作起来直截了当。Cassandra 就是一种分布式键值数据库，它以简单和可测量性为设计理念。尽管经常拿它与 HBase（详细描述见本章后面）相提并论，但是Cassandra 还是有一个关键的不同之处：

- Cassandra 是一个全包容系统，意思是它不需要一个 Hadoop 环境或者任何其他的大数据工具。
- Cassandra 是完全无主节点的，它的运算就是一个对等计算（peer-to-peer, P2P）系统。这样使得它更容易配置，且具有高弹性。

2.1.1 教程链接

DataStax 是一个对 Cassandra 提供商业支撑的公司，提供了一些免费可用的视频（*http://bit.ly/1DhUA7t*）。

2.1.2 示例代码

与 Cassandra 进行交互最简单的方式就是通过它的 shell 接口，可以通过运行安装目录下的 bin/cqlsh 来开始 shell。

接下来需要创建一个 keyspace。keyspace 和传统的关系型数据库中的 schema 很相似，该方法对组织 table 非常实用。一个典型的示例是为每一个应用程序使用一个独立的不同的 keyspace：

```
CREATE KEYSPACE field_guide
WITH REPLICATION = {
    'class': 'SimpleStrategy', 'replication factor' : 3 };

USE field_guide;
```

通过以上代码，现在就有一个 keyspace 了，在这个 keyspace 上通过创建一个 table 可以保存评论。这个 table 将有 3 列和一个主关键字，该主关键字由 reviewer 和 title 组成且它们在数据库中是独一无二的。

```
CREATE TABLE reviews (
    reviewer varchar,
    title varchar,
    rating int,
    PRIMARY KEY (reviewer, title));
```

一旦 table 被创建，就可以插入一系列的评论了：

```
INSERT INTO reviews (reviewer,title,rating)
    VALUES ('Kevin','Dune',10);
INSERT INTO reviews (reviewer,title,rating)
    VALUES ('Marshall','Dune',1);
INSERT INTO reviews (reviewer,title,rating)
    VALUES ('Kevin','Casablanca',5);
```

到现在，已经有许多数据了，你将需要创建一个 index，有了它可以通过执行一个简单的 SQL 查询来检索出 *Dune* 的评论：

```
CREATE INDEX ON reviews (title);

SELECT * FROM reviews WHERE title = 'Dune';

reviewer |      title | rating
---------+------------+-------
   Kevin |       Dune | 10
Marshall |       Dune | 1
   Kevin | Casablanca | 5
```

2.2 HBase

许可证	Apache License, Version 2.0
活跃度	高
用途	可随机访问的 NoSQL 数据库
官方网站	https://hbase.apache.org
Hadoop 集成度	完全集成

在很多情况下，你的数据量可能非常少。也就是说，虽然数据有许多属性，但是每一次要查看的都只占其中的一小部分。举个例子，你也许想得到包含一个帮助应用程序中各种各样 ticket 的 table，电子邮件上的 ticket 也许与网络问题如丢失密码或备份系统问题上的 ticket 的信息（属性或列）大不相同。还有另一种情况，就是数据在一列或一个属性上有大量的相同值，叫作“country”或“state”。当遇到这样的例子时，尝试着去考虑 HBase。

HBase 是一个 NoSQL 数据库系统，包含在标准的 Hadoop 发行版本中。它是一种逻辑上的键值存储。它的意思是，行由一个 key 来定义，且与它们相关的 value 在大量的二进制文件（或列）中被存储起来。唯一的数据类型就是字节串。物理上相近的列以家庭组的形式被存放在一起。很多时候，HBase 通过 Java 代码来进行访问，但是存在一些通过 Pig、Thrift、 Jython (基于 Python), 以及其他一些工具来使用 HBase 的 API。HBase 通常并不是按 MapReduce 的方式访问。它是通过一个 shell 接口来交互使用的。

HBase 经常用在那些可能需要较少行的应用程序中。也就是说，每一行也许只用到已定义的少数几列。（随着 hadoop）通过主关键字或者定义的键值来完成元素的访问是非常快速的。它有高度可伸缩性且反应迅速。与传统的 HDFS 应用程序不同，它允许随机访问行，而不需要顺序检索。

尽管速度比 MapReduce 快，但是不能将 HBase 用于任何一种事务性的需求，也不能用于任何一种关系分析。因为它不支持任何形式的二级检索，所以找到具有

特定值的所给列的所有行是非常冗长的，并且该过程必须在应用层完成。HBase没有JOIN操作，这个处理必须通过特定的应用程序来完成。你必须为应用层提供安全，其他的一些诸如Accumulo（在本章后面有详细描述）的工具也要本着安全的理念来构建。

尽管在今天，或许Cassandra（在本章前面有详细描述）和MongoDB（在本章后面有详细描述）仍然是NoSQL数据库的主力军，但是随着HBase的逐渐流行，也许在不远的未来，HBase将成为NoSQL的领军力量。

2.2.1 教程链接

人们在Coreservlets.com“http://www.coreservlets.com”上放置了一些非常好的HBase系列（*http://bit.ly/1zWjIhF*）的Hadoop教程。在因特网上（*http://youtu.be/IumVWII3fRQ*）也发现一些对我们特别有帮助的十分有用的视频教程。

2.2.2 示例代码

在这个例子中，目标是找到对电影*Dune*评价的平均值。每一部电影的评价包括3个元素：评价者的名字、电影的标题，以及评价等级（从0到10的整数）。该例子用HBase shell完成如下：

```
hbase(main):008:0> create 'reviews', 'cf1'
0 row(s) in 1.0710 seconds

hbase(main):013:0> put 'reviews', 'dune-marshall', \
hbase(main):014:0> 'cf1:score', 1
0 row(s) in 0.0370 seconds

hbase(main):015:0> put 'reviews', 'dune-kevin', \
hbase(main):016:0> 'cf1:score', 10
0 row(s) in 0.0090 seconds

hbase(main):017:0> put 'reviews', 'casablanca-kevin', \
hbase(main):018:0> 'cf1:score', 5
0 row(s) in 0.0130 seconds

hbase(main):019:0> put 'reviews', 'blazingsaddles-bOb', \
hbase(main):020:0> 'cf1:score', 9
0 row(s) in 0.0090 seconds

hbase(main):021:0> scan 'reviews'
ROW                          COLUMN+CELL
 blazingsaddles-bOb          column=cf1:score,
                             timestamp=1390598651108,
                             value=9
 casablanca-kevin            column=cf1:score,
```

```
                              timestamp=1390598627889,
                              value=5
 dune-kevin                   column=cf1:score,
                              timestamp=1390598600034,
                              value=10
 dune-marshall                column=cf1:score,
                              timestamp=1390598579439,
                              value=1
3 row(s) in 0.0290 seconds

hbase(main):024:0> scan 'reviews', {STARTROW => 'dune', \
hbase(main):025:0> ENDROW => 'dunf'}
ROW                           COLUMN+CELL
 dune-kevin                   column=cf1:score,
                              timestamp=1390598791384,
                              value=10
 dune-marshall                column=cf1:score,
                              timestamp=1390598579439,
                              value=1
2 row(s) in 0.0090 seconds
```

现在通过扫描一个高效的范围得到了两行数据，但是如何来计算平均数呢？在HBase shell里，这是不可能的。使用HBase Java API，可以提取值，但是这里没有内置的行内聚功能来计算平均数或求和，所以需要通过Java代码来完成这件事情。

在HBase上选择行关键字是非常有讲究的。如果想得到Kevin对所有电影评价的等级的平均值，那么需要一个完整的table扫描，对于一个非常大的数据集来说这可能是一项非常冗长的工作。此时你也许想拥有两个版本的table，一个行关键字由评价者—电影给出，而另外一个由电影—评价者给出。那么接下来的问题就是确保它们同步了。

2.3 Accumulo

许可证	Apache License, Version 2.0
活跃度	高
用途	基于 cell-level 安全的名 - 值数据库
官方网站	http://accumulo.apache.org/index.html
Hadoop 集成度	完全集成

假如有一个应用程序，它使用了像 HBase（在本章前面有详细描述）优秀的列 / 名 - 值存储，但是还有一个附加的安全问题，你必须要非常小心地控制那些使用者可以看到数据单元。举个例子，有一个多用户数据存储，在它上面可以将企业中不同部门的数据存放在一个单独的数据表上，并且要确保一个部门中的用户不能看到其他部门的数据，但是高级管理员可以看到整个企业的数据。处于这样一个内部安全原因，美国国家安全局（NSA）开发了 Accumulo，然后将其代码捐赠给 Apache 基金会。

你可能会注意到 HBase 和 Accumulo 之间有很多相似的地方，这两个系统都是在 Google 的 BigTable 上进行建模的。Accumulo 着眼于安全和基于单元的访问控制，对模型进行了改善。每一个用户都有一套安全标签，即简单的文本字符串。假设你是“admin”，“audit”或是“GroupW”。当你想去定义一个特殊单元的访问时，你可以将那些给出行的列的可见性设置成各种各样的标签形式的布尔表达式。在这种规则下，& 是逻辑与，| 是逻辑或。如果单元的可见性规则是 admin|audit，就意味着任何用户无论是 admin 标签还是 audit 标签都可以看到这个单元。如果单元的可见性规则是 admin&Group7，那么这两个标签都是需要的，如果缺少 Group7 标签将看不到这个单元。

Accumulo 不仅具有安全性，它同样可以以大规模的形式运行，伴随着 PB 级的数据，以及每秒成百上千的存储及检索操作。

2.3.1 教程链接

更多有关 Accumulo 的信息，请查看以下的资源：

- 来自 Aaron Cordova 的介绍（*http://slidesha.re/1E9zxlK*），Accumulo 的一个发起者。
- 一个关注于性能和 Accumulo 风格的视频教程（*http://youtu.be/5RPaDzOwqQ8*）。
- 一个更加关注安全和加密的教程（*http://youtu.be/71J65mKm6ZU*）。
- Accumulo 顶级会议（*http://bit.ly/1zvSXwT*）上也包含大量丰富的信息。

2.3.2 示例代码

一个好的示例代码放在这里显得过长且过于复杂，但是可以在产品主页的“Examples”这部分（*http://bit.ly/16TINzP*）找到相应的内容。

2.4 Memcached

许可证	Revised BSD License
活跃度	中
用途	在内存上缓存
官方网站	http://memcached.org
Hadoop 集成度	不整合

很有可能在某一天你会遇到这样一种情况，就是需要在很短的时间里非常快速地访问大量的数据。举个例子，比如说想给客户和潜在客户发邮件，让他们知道有关产品新增的一些新特性，但是同时也需要排除那些在这个月已经联系过的客户。

在大数据系统中进行查询的典型方法是将庞大的联系列表分布在许多机器上，然后将这个月中已经联系过的全部客户列表加载到每个机器的内存上，并且对照已经发过邮件的客户列表对每一个联系人进行快速对比检测。在 MapReduce 上，这个通常被称为“replicated join”。但是，让我们假设你从展销会、产品演示，以及社交媒体上得到了一个庞大的由数百万电子邮件地址组成的联系网络，且希望相当频繁地去联系这些人。这就意味着那些这个月已经联系过的人的列表也许相当庞大，完整的列表可能无法放在每台机器的可用内存中。

那么你真正需要的方法是组成一个跨越所有机器的内存池，并将东西都放在这个庞大的内存池中。Memcached 就是这样一个工具，它可以帮助你构建这样一个分布式的内存池。重新回到我们前面所提的例子上，你可以将那些已经发过邮件的客户的整个列表存储在分布式内存池中，并且引导所有不同的机器将完整联系列表提交给内存池而不是逻辑内存。

2.4.1 教程链接

Spymemcached 项目（*http://bit.ly/1zLRaZC*）提供了几个使用 API 的可用实例，该项目的维基百科地址是 *http://bit.ly/1KMwgv8*。

2.4.2 示例代码

假设我们需要记录哪些评论者已经评价过哪些电影，因为我们不能要求一个评论者去评价同样的电影两次。因为没有一个支持 Memcached 的独立的、正式的 Java 客户端，所以我们将使用流行的 spymemcached 客户端。

从定义一个 client 开始，并将它指向我们的 Memcached 服务器：

```
MemcachedClient client = new MemcachedClient(
    AddrUtil.getAddresses("server1:11211 server2:11211"));
```

现在将数据加载到缓存中，我们将使用流行的 OpenCSV library "*http://opencsv.sourceforge.net*" 来读取评论者文件，然后将找到的每一个评论者及标题对写入缓存中：

```
CSVReader reader = new CSVReader(new FileReader("reviews.csv"));
String [] line;
while ((line = reader.readNext()) != null) {
    // 合并评论者姓名和电影标题
    // 放到一个单独的 value (ie: KevinDune)
    // 可以将它作为关键字来使用
    String reviewerAndTitle = line[0] + line[1];
    // 将这个关键字写入缓存中并存储 30 分钟
    //(188 秒)
    client.set(reviewerAndTitle, 1800, true);
}
```

一旦将值加载到缓存中，我们就可以快速地通过一个 MapReduce 工作或者其他的 Java 代码对缓存进行检测：

```
Object myObject=client.get(aKey);
```

2.5 Blur

许可证	Apache License, Version 2.0
活跃度	中
用途	文档仓库
官方网站	https://incubator.apache.org/blur
Hadoop 集成度	完全集成

假设你已经购买了使用 Hadoop 的一整套大数据工具。你通过使用 Flume 来搜集数据并将它们放到 HDFS 上，你的 MapReduce 工作将对这些数据进行转化并构建键值对，并将这些键值对放到 HBase 上，甚至一些有进取心的数据科学家可以帮助你通过 Mahout 来分析你的数据。而此时此刻，你的 CTO 走到你面前然后询问你，通过那些收集到的使用者的反馈表格是否能得到你想要的某个特定产品被提及的频率。此时你可能会很沮丧，因为你意识到反馈的只是一种自由格式的文档，你根本就没有办法在上面寻找任何数据。

Blur 就是一种在 Hadoop 上索引和检索文档的工具。因为它以 Lucene（一个非常流行的 text-indexing 框架）为内核，它有非常多有用的特性，包括模糊匹配、通配符搜索以及页面反馈。它为你提供了一种在非结构化数据上进行检索的方法，如果没有这种方法，此类检索将会变得非常复杂。

2.5.1 教程链接

通过学习项目主页（*http://bit.ly/1CgXgRf*）上官方的“getting started”指南是一个不错的开始。虽然在 Hadoop User Group 会议上展示（*http://youtu.be/w4zLz9ussdI*）的内容不是很多，但它却是一个非常好的学习地方。

2.5.2 示例代码

这里有两种不同的将数据加载到 Blur 上的方法。当有大量数据要做大量的索引时，更适合使用 MapReduce。反之，如果想用流数据，最好使用变形（适配）接口。在这个例子中，我们使用变形（适配）接口，因为我们仅仅需要给一对记录做索引。

```
import static org.apache.blur.thrift.util.BlurThriftHelper.*;

Iface aClient = BlurClient.getClient(
    "controller1:40010,controller2:40010");

// 在 table1 上创建一个新的行
RowMutation mutation1 = newRowMutation("reviews", "Dune",
    newRecordMutation("review", "review_1.json",
        newColumn("Reviewer", "Kevin"),
        newColumn("Rating", "10")
        newColumn(
            "Text",
            "I was taken away with the movie's greatness!")
    ),
    newRecordMutation("review", "review_2.json",
        newColumn("Reviewer", "Marshall"),
        newColumn("Rating", "1")
        newColumn(
            "Text",
            "I thought the movie was pretty terrible :(")
    )
);

client.mutate(mutation);
```

现在假设我们要在评论文本中检索那些提到 great 的所有评论。我们将通过运行安装目录里的 */bin/blur shell* 来加载 Blurshell 并运行一个简单的查询。这个查询告诉 Blur 在 reviews 表中的 review 的列成员中的“Text”列中查找类似单词“great”的内容。

```
blur> query reviews review.Text:great
 - Results Summary -
    total : 1
    time  : 41.372 ms
-------------------------------------------------------------
      hit : 0
    score : 0.9548232184568715
       id : Dune
 recordId : review_1.json
   family : review
     Text : I was taken away with the movie's greatness!
-------------------------------------------------------------
- Results Summary -
    total : 1
    time  : 41.372 ms
```

2.6 Solr

许可证	Apache License, Version 2.0
活跃度	高
用途	文档仓库
官方网站	https://lucene.apache.org/solr
Hadoop 集成度	API 兼容

在很多时候，仅仅是想在一大堆文档中找到某个东西，并不是所有的工作都需要跨越 PB 级的数据而进行庞大复杂的分析工作。对于很多常见的例子，你可能会发现，对于一些简单的 grep UNIX 或 Windows 搜索来说，你的数据量太大，但是又不足以让专门的数据科学家团队来做这些事情。Solr 就适合这样的中间情况，它提供了一个易于使用的方法，可以实现快速索引和搜索大量文档中的内容。

Solr 支持分布式体系结构，这将带来许多你所期望的能从大数据系统中获得的好处（比如，线性扩展、数据复制及失效备援）。它是基于 Lucene（一个很受欢迎的索引和文档搜索框架）的，并且通过提供一整套工具实现了该框架，建立索引和查询数据。

虽然 Solr 能够使用 Hadoop 分布式文件系统 (HDFS，在第 1 章有详细描述）来存储数据，但是它不是真正的和 Hadoop 兼容，并且它也不是通过使用 MapReduce(在第 1 章有详细描述) 或者 YARN(在第 1 章有详细描述) 来建立索引和对检索做出响应的。与它有一个类似的产品名字叫 Blur(在第 2 章有详细描述)，在 Lucene 框架上利用整个 Hadoop 堆栈构建工具。

2.6.1 教程链接

除了 Solr 官方主页上的教程，在 Solr 维基百科（*http://bit.ly/199s7Wh*）上也有大量的信息。

2.6.2 示例代码

在这个例子中，假设我们有一组由电影评论标签组成的半结构化数据，这些数据有非常清晰的标题和评论文本标识。这些评论将被存储在 *reviews* 目录下的独立的 JSON 文件中。

首先我们将告诉 Solr 来对数据进行索引，有少数的不同的方法可以做到这点，它们都有着各自特有的权衡。在这个例子中，我们将使用最简单的途径，就是使用位于 Solr 安装目录下的子目录 *exampledocs* / 中 *post.sh* 脚本。

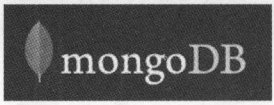
```
./example/exampledocs/post.sh /reviews/*.json
```

一旦为我们的评论建立起了索引，它们就做好检索的准备了。Solr 拥有自己的图形用户界面 (GUI)，该用户界面可被用于简单的搜索。我们使用这个 GUI 来搜索电影评论中包含单词“great”的内容：

```
review_text:great&fl=title
```

这个查询告诉 Solr，我们想要在 `title` 的领域 (`fl=title`) 中检索出那些在 `review_text` 中出现单词“great”的所有评论。

2.7 MongoDB



许可证	Free Software Foundation's GNU AGPL v3.0.; commercial licenses available from MongoDB, Inc.
活跃度	高
用途	JSON 面向文档型数据库
官方网站	http://www.mongodb.org
Hadoop 集成度	API 兼容

如果在 Hadoop 集群上有大量的 JSON 文档，并且需要一些数据管理工具来有效地使用它们，这时就可以考虑 MongoDB 了，它是一种开源的、大数据，以 JSON 格式为目标的面向文档的数据库。在 2015 年初，它是一个最受欢迎的 NoSQL 数据库。与其他数据库系统不同，MongoDB 支持二级索引，这就意味着它可以实现快速的检索，而不同于在 Mongo 的数据库上用主键来唯一标识每一个文档。这个名字来源于俚语词“humongous”，意思是非常、非常巨大。尽管 MongoDB 起初是没有在 Hadoop 和 HDFS 上运行的，但是它可以和 Hadoop 一起使用。

MongoDB 是一个面向文档型数据库，文档是一个 JSON 对象。在关系型数据库中，你有表格和行。在 MongoDB 中，一行相当于一个 JSON 文档，一系列的 JSON 文档聚在一起就类比成一个表格。为了理解 MongoDB，你可以先跳到本书的第 3 章提前阅读“JSON”。

或许理解 MongoDB 使用的最好方式就是通过运行代码示例获知，你将会在接下来的“示例代码”这一部分中看到。

2.7.1 教程链接

项目官方页面上的教程部分（*http://bit.ly/1zWkb3B*）不愧是一个开始 MongoDB 的好地方。在互联网上也有许多很好的可用的视频资源，如 *http://youtu.be/liQzIsFnCr0*。

2.7.2 示例代码

现在你将在标准的数据集中计算电影 *Dune* 的平均排名。如果你了解 Python 语言，将会非常清晰。如果你不了解，代码仍然是相当简单的：

```
#!/usr/bin/python
# 输入所需包
import sys
import pymongo

# json 电影评论
movieReviews = [
    { "reviewer":"Kevin", "movie":"Dune", "rating","10" },
    { "reviewer":"Marshall", "movie":"Dune", "rating","1" },
    { "reviewer":"Kevin", "movie":"Casablanca", "rating","5" },
    { "reviewer":"Bob", "movie":"Blazing Saddles", "rating","9" }
]

# MongoDB 链接 info
MONGODB_INFO = 'mongodb://juser:password@localhost:27018/db'

# 链接 MongoDB
client=pymongo.MongoClient(MONGODB_INFO)
db=client.get_defalut_database()

# 创建一个电影链接
movies=db['movies']

# 插入电影评论
movies.insert(movieReviews)

# 通过遍历，找到所 title 为 Dune 的电影，
# 找到所有的分数，
# 使用标准数据库游标技术
mcur=movies.find({'movie': {'movie': 'Dune'})
count=0
sum=0

# 对于所有 Dune 的评论，计算它们的总数然后求排名总和
for m in mcur:
    count += 1
    sum += m['rating']
client.close()
rank=float(sum)/float(count)
print ('Dune %s\n' % rank)
```

2.8 Hive

许可证	Apache License, Version 2.0
活跃度	高
用途	数据交互
官方网站	http://hive.apache.org
Hadoop 集成度	完全集成

首先，在 Hadoop 集群上的所有数据的访问都是通过由 Java 编写的 MapReduce 作业来完成的。如果所有的 Hadoop 用户都有一个稳定的 Java 语言的程序员，在 Hadoop 初期这个方法是非常管用的。但是，随着 Hadoop 的发展，Hadoop 出现到了更为广阔的世界里，很多地方想使用 Hadoop，但是那些熟练的 SQL 程序员对于写 MapReduce 代码可能会感到非常困难。进入 Hive，Hive 的目的就是允许 SQL 能访问 HDFS 中的数据。Apache Hive 数据仓库软件使查询和管理 HDFS 上的大数据集变得非常容易。Hive 定义了一个简单的类 SQL 查询语言，叫作 HQL，这种语言让用户能用熟悉的 SQL 来查询数据。用 HQL 编写的查询语句通过 Hive 转化成 MapReduce 代码并且通过 Hadoop 来执行。但是要注意！ HQL 并不是完全 ANSI 标准的 SQL。尽管一些基本的性能都有，但是还是有一些特性丢失了。这里是当前年初列出的部分列表：

- Hive 不支持不等连接条件。
- 不支持 update 和 delete 语句。
- 不支持事务。

你也许不需要用到这些，但是如果你运行的代码是由第三方解决方案生成的，它们就有可能会生成一些 Hive 不兼容的代码。

Hive 不要求数据以“Hive 形式”读或写，根本也就没有这样的东西。这就意味着你的数据可以直接由 Hive 访问，而不需要做任何的提取转换加载（ETL）预处理，

就像典型的传统关系数据库所需要的那样。

2.8.1 教程链接

有几个很好的资源分别是Hive教程官网（*http://bit.ly/1KJbmih*）和在HortonWorks上人们发布的视频资料（*http://youtu.be/Pn7Sp2-hUXE*）。

2.8.2 示例代码

有一个逗号分隔值（CSV）的文件包含有关评论者，电影以及评价等级电影评论信息：

```
Kevin,Dune,10
Marshall,Dune,1
Kevin,Casablanca,5
Bob,Blazing Saddles,9
```

首先，需要定义数据的模式：

```
CREATE TABLE movie_reviews
    ( reviewer STRING, title STRING, rating INT)
ROW FORMAT DELIMITED
FiLEDS TERMINATED BY  '\,'
STORED AS TEXTFiLE
```

接下来，需要通过电影评论文件中所指的表格加载数据。因为Hive不需要这些数据以任何特定的格式存储，所以加载一个由HDFS上Hive简单指向的文件组成的表格：

```
LOAD DATA LOCAL INPATH  'reviews.csv'
OVERWRITE INTO TABLE movie_reviews
```

现在已经准备好去执行某些类型的分析。在这个例子中，假设想要找到电影Dune的平均等级：

```
Select AVG(rating) FROM movie_reviews WHERE title =  'Dune' ;
```

2.9 Spark SQL (前身是 Shark)

许可证	Apache License, Version 2.0
活跃度	高
用途	SQL 访问 Hadoop 上的数据
官方网站	http://spark.apache.org/sql/
Hadoop 集成度	API 兼容

如果你需要用 SQL 来访问你的数据，但 Hive(在本章前面有详细描述) 有点表现不佳，并且你又愿意将其提交给 Spark(在第 1 章有详细描述) 环境，那么你不妨考虑一下 Spark SQL。最初用 SQL 在 Spark 上进行访问被称为 Shark 工程，且是 Hive 的一个端口，但是后来 Shark 已经停止发展，而它的继承者 Spark SQL 现在已经是 Spark 上的主流 SQL 工程。博客“Shark, Spark SQL, Hive on Spark, and theFuture of SQL on Spark”（*http://bit.ly/1AmBixa*）给出了更多有关这些变化的信息。Spark SQL 和 Spark 一样，有一个内存计算模型，它可以有助于提高速度。仅仅是在最近几年因为内存成本的降低导致大内存 Linux 服务器无所不在，从而使得大数据集的内存计算的快速发展。因为内存的访问时间通常比磁盘的访问要快 10 倍，所以尽可能多地在内存中存放数据而尽可能少地使用磁盘是非常吸引人的。但是放弃 MapReduce 会使 Spark SQL 更快，即使它需要进行磁盘访问。

尽管 Spark SQL 用一种 Hive 查询语言 HQL 来表达，但是它也有一些在 Hive 上没有的额外功能。一个就在整个用户会话期间 encache 表数据的能力。这相当于其他一些数据库中的临时表，但又和其他一些数据库不同，这些表存在于内存中，因此访问速度要快得多。SparkSQL 还允许类似 Spark 弹性分布式数据集（RDD）那样访问表格。

Spark SQL 支持 Hive 元存储、大部分的查询语言，以及数据格式，所以现有的 Hive 用户应该向 Shark 转换比向其他工具转换更为容易。但是，尽管现有的 Spark SQL 文档在这点上并不是完全说明，但并不是所有的 Hive 功能都没有被 Spark SQL 实现。对 Python, Java, 和 Scala 已经有 API 了。可以阅读本章前面的

“Hive”来获取更多的详细内容。Spark SQL 同样也可以运行 Spark MLlib 的机器学习算法，就好像 SQL 语句一样。

Spark SQL 可以使用 JSON(在第 3 章有详细描述)和 Parquet(在第 3 章有详细描述)作为数据资源，所以它在 HDFS 环境中是非常有用的。

2.9.1 教程链接

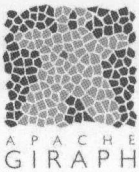
在项目主页上有很多非常有价值的教程。（*http://bit.ly/1E9AmLj*）。

2.9.2 示例代码

在用户层面上，Shark 和 Hive 看起来很像，所以如果你能够在 Hive 上编程，你差不多也能在 Spark SQL 上编程。但是你需要安装你的 Spark SQL 环境。在这里你将使用和其他例子一样的电影评论数据，以及在 Python 上能用这些数据做什么（要理解安装过程，你可以阅读本书第 1 章的“Spark”，还有那些有关 Python 的知识）：

```
# Spark 需要一个 Context 对象。让我们假设它已经存在。
# 你同样需要一个 SQL Context 对象
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)

# 加载一个 CSV 文本文件并且将它的每一行转化为 Python
# 字典为匿名函数使用 lambda 符号
lines = sc.textFile("reviews.csv")
movies = lines.map(lambda l: l.split(","))
reviews = movies.map(
    lambda p: {"name": p[0], "title": p[1], "rating": int(p[2])})

# Spark SQL 将 RDD 看成是一个数据模型且注册表的名字
schemaReviews = sqlContext.inferSchema(reviews)
schemaReviews.registerAsTable("reviews")
# 一旦你将 RDD 注册为一个模型，你就可以在它上面运行 SQL 语句了
dune_reviews = sqlContext.sql(
    "SELECT * FROM reviews WHERE title = 'Dune'")
```

2.10 Giraph



许可证	Apache License, Version 2.0
活跃度	高
用途	图表数据库
官方网站	https://giraph.apache.org
Hadoop 集成度	完全集成

你也许知道一个称为 6 度分割理论的室内游戏，它来自凯文·贝肯，在他的电影花絮中，专家试图找到电影演员和凯文·贝肯之间的关系。如果一名演员在同一部电影中出演，那么“路径”的长度为 1。如果一个演员和凯文·贝肯从来没有在一部电影中出演，但是和一个与凯文·贝肯在同一部电影中出现过的演员在同一部电影中出演，则它的路径的长度是 2。建立假设在电影界中的任意一个个体都可以在 6 步或 6 度分割内通过他或她的电影角色与凯文·贝肯产生联系。举个例子，在凯文·贝肯和肖恩·潘之间有一道弧，因为他们都在电影《神秘河》中，所以他们之间有一个 1 度分割或者说他们之间的路径长度为 1。但是本尼西奥·德尔·托罗与凯文·贝肯之间的路径长度为 2，因为他从来没有和凯文·贝肯在同一部电影中出演过，但是他和肖恩·潘在同一部电影中出演过。

用一组有序对（N,M）来描述一个从 N 到 M 的连接，你可以通过这些关系图来展示人际关系。

你可以将一棵树（如分层文件系统）看成是一个关系图，它有一个开始节点，然后由弧引出树的分支。集合 {(top, b1), (top, b2), (b1,c1), (b1,c2), (b2,c3)} 是从树的根部，通过树枝从 top 连接到 b1，b2，然后再连接 b1 到 c1 及 c2，b2 到 c3。元素集 {top, b1, b2, c1,c2,c3} 则称为节点。

你会发现用这个关系图来描述实体之间的关系是非常有用的。举个例子，如果你有一个公司里的电子邮件的发送集合，你可以建立一个公司人员关系图，每一个

节点代表一个人，而如果 a 给 b 发过一封邮件，或者 b 给 a 发过一封邮件，则存在一个从 a 到 b 的弧。它画出来就像图 2-2 这样。

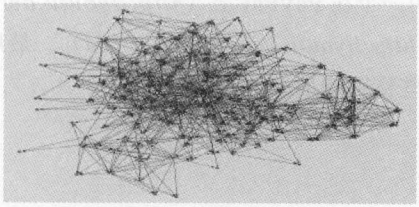


图 2-2: 人与人之间的邮件关系图

Giraph 是一个 Apache 项目，它可以通过关系图构建和提取信息。举个例子，你可以用Giraph来计算关系图中一个节点到另一个节点的最短路径（连接弧的个数）或者来判断两个节点中是否有路径。

Apache Giraph 来自一个叫作 Pregel 的 Google 项目，被用在 Facebook 中来建立和分析万亿节点之间的关系图，不可否认它在一个非常大的 Hadoop 集群中。它的建立使用了一种称为整体同步并行计算模型（Bulk Synchronous Parallel，BSP）的技术。

一般来说，在 BSP 模型中有一系列的“超步”。在零步时，将顶点或节点分发给工作流程。在接下来的每一个超步中，每一个顶点通过一组信息迭代，这组信息从之前的超集接收且发送给与它有联系的其他节点信息。

在凯文·贝肯的例子中，每一个节点代表一个演员，导演，制片人，编剧等，每一个弧将同一部电影中共事的两个人连接在一起。我们想对刚才的假设进行测试，就是这个行业表示的关系图中的每一个人于凯文·贝肯之间的孤的段数都不会超过 6。每一个节点都给定一个分段个数的初始值，凯文·贝肯就是 0。对于其他人，这个初始值是一个非常大的整数。在第一超步中，每一个节点将它的值发送给所有与它有联系的其他节点。接下来，在其他的超步中，每一个节点首先读取这所有的信息，并将保留最小值。如果小于当前值，则节点加 1，然后在该超步结束时将该值发送给所有与它相连的节点。

为什么？因为如果一个已连接过的节点到凯文·贝肯有 N 步，则这个节点最多离凯文·贝肯有 N+1 步远。一个节点一旦建立了新的值，它会有选择的发出更多的信息。

在第 6 超步的最后，所有的人都在 6 步或者更少的步数内与凯文·贝肯相连。

2.10.1 教程链接

产品的官方页面有一个快速开始向导（*http://bit.ly/1vkfaxD*）。此外，有少许一些的谈话录像，其中一个在 PayPal 上（*http://youtu.be/_xha7OdEy2A*），另一个在 Facebook 上（*http://youtu.be/b5Qmz4zPj-M*）。最后，在微博 (*http://bit.ly/1DhV1i3*) 上还有一些特别的信息。

第 3 章

序列化

大数据系统花费了大量的时间和资源用于数据移动。举一个典型的查看日志过程的例子。这个过程可能会从其他好几个服务器上搜集日志，将这些日志放到 HDFS 上，并通过执行某种形式的分析来产生一些报告，然后将这些报告放到一些仪表板上让用户可以看到。在这个过程的每一步中，在某些情况下，你需要多次地在各系统之间，将数据从硬盘驱动器移动到内存中，如图 3-1 所示。

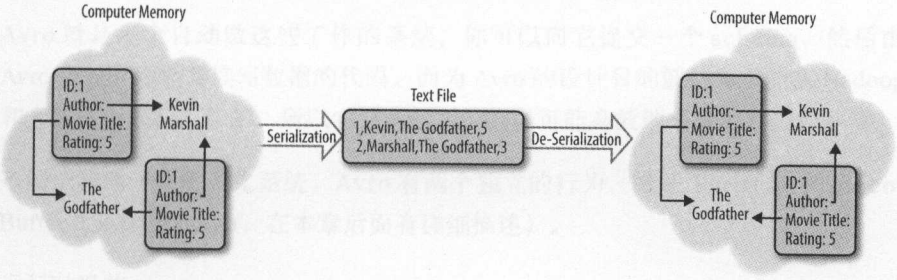


图 3-1：一个电影评论的序列化和反序列化

用现代计算机来处理数据，它通常有各种复杂的格式、完整的内部关系和引用。当你想将这些数据记录下来，无论是分享它，还是在稍后存储它，你都需要找到一个分解这些关系、解释这些引用，并建立一个从开始到结束都可以读取的数据表示的方法。这个过程就叫作序列化。

同样的，你有没有这样一种感受，就是通过阅读一个地方或者事件的详细描述，你就可以在你的脑海中将其完美地呈现出来？这种读取事物的过程就是记录的过程，也就是序列化，而重建这些复杂的引用以及关系的过程被称为反序列化。

有各种各样的数据序列化工具和框架可以帮助管理那些看起来好像在移动的数据。为每一个任务选择一个正确的序列化格式是构建一个具有可扩展性、运行良好，且易管理的系统的关键方面。你将看到，在选择一个序列化格式所要考虑到的少数几个重要方面：

数据量

数据占用的内存或磁盘的空间大小。

读 / 写速度

计算机读 / 写数据时所需要的时间。

可读性

人们在没有外界帮助的情况下是否可以理解序列化数据？

易用性

这个格式写或读数据的难易程度。当其他人想读取你的数据时，是否需要向他们提供专门的文件或者工具。

3.1 Avro

许可证	Apache License, Version 2.0
活跃度	中
用途	数据序列化
官方网站	http://avro.apache.org
Hadoop 集成度	API 兼容

如果你有很多数据，并且想和他人分享它们。首先你要做的就是描述出这些数据的数据结构，比如定义这些数据有多少字段，而每个字段又包含些什么样的数据类型。用专业术语来描述，称该定义为一个 schema。你可能会将 schema 连同你的数据一起与他人进行分享，如果有人对你的数据感兴趣可能会通过一小段代码来确保他们可以读它。

Avro 就是一个自动做这些工作的系统。你可以向它提交一个 schema，然后由 Avro 来建立你需要读写数据的代码。因为 Avro 的设计目的就是为了能在 Hadoop 和大数据上进行工作的，所以它会竭尽所能的尽可能高效地来对数据进行存储。

不同于其他一些序列化系统，Avro 有两个独立的行为，就是 Thrift 和 Protocol Buffer（简称 protobuf，在本章后面有详细描述）。

运行时组装

Avro 不需要生成特有的序列化代码以及预先共享。这简化了跨越多个平台部署应用程序的过程，但这是以性能为代价的。在有些时候，你可以通过预先生成代码来解决这个问题，但是你需要在每次改变数据格式的时候重新生成和共享这些代码。

Schema 驱动

每一个数据的传输包括两个方面：schema 描述数据格式和数据本身。因为在 schema 中定义了数据格式，每一个元素都不再需要被标记。这样大大减少了传输复杂对象的开销，但是实际上另一方面它也增加了传输大量的但是简单

的对象的开销。

3.1.1 教程链接

官方的 Avro 文档网页（*http://bit.ly/1CVll30*）是一个很好的开始学习的地方，它同时提供了 Java 和 Python 语言的“入门指南”。如果你对将 Avro 集成到 MapReduce 上的讲解有更多的兴趣的话，GitHub 上的 avro-mr-sample 项目（*http://bit.ly/1FyNXNJ*）绝对是一个不错的选择。

3.1.2 示例代码

Avro 支持两种通用模型：

- 一种是传统的序列化模型，在这个模型上，开发人员发布一个 schema，基于这个 schema 通过运行编译程序来创建模型，然后将这些模型用在他们的应用程序上。
- 一种是运行时模型，在这个模型上，Avro 基于由一个运行时提供的 schema 文件来建立记录。

在下面的例子中，我们将使用运行时模型因为它是 Avro 最有趣的差异之一。

首先我们定义一个 schema 文件，我们称它为 *review.avsc*：

```
{"namespace": "example.elephant",
 "type": "record",
 "name": "Review",
 "fields": [
    {"name": "reviewer", "type": "string"},
    {"name": "movieTitle", "type": "string"},
    {"name": "rating", "type": "int"}
 ]
}
```

现在我们可以通过这个 schema 来创建一个对象，并且将它写到磁盘上：

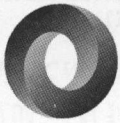
```
// 绑定 schema
Schema schema = new Parser().parse(new File("review.avsc"));

// 建立一个记录
GenericRecord review = new GenericData.Record(schema);
review.put("reviewer", "Kevin");
review.put("movieTitle", "Dune");
review.put("rating", 10);

// 序列化评论并写到磁盘
File file = new File("review.avro");
DatumWriter<GenericRecord>datumWriter =
```

```
        new GenericDatumWriter<GenericRecord>(schema);
    DataFileWriter<GenericRecord>dataFileWriter =
        new DataFileWriter<GenericRecord>(datumWriter);
    dataFileWriter.create(schema, file);
    dataFileWriter.append(user1);
    dataFileWriter.append(user2);
    dataFileWriter.close();
```

我们刚刚创了一个充满评论对象的文件，我们同样也可以对该文件进行反序列化：

```
    // 绑定 schema
    Schema schema = new Parser().parse(new File("review.avsc"));

    File file = new File("review.avro");
    DatumReader<GenericRecord>datumReader =
        new GenericDatumReader<GenericRecord>(schema);
    DataFileReader<GenericRecord>dataFileReader =
        new DataFileReader<GenericRecord>(file, datumReader);
    GenericRecord review = null;

    while (dataFileReader.hasNext()) {
    // 通过 next() 来重用用户对象，这可以节省那些在很多项目中都有的
    // 分配和无用信息收集的对象文件
        review = dataFileReader.next(review);
    }
```

3.2 JSON



许可证	http://www.json.org/license.html
活跃度	中
用途	数据描述和传递
官方网站	http://www.json.org
Hadoop 集成度	不集成

因为 JSON 并不是 Hadoop 的一部分，所以你可能会想知道为什么我们要在这里介绍它。在 Hadoop 上，JSON 变得越来越普遍，因为它实现了各领域的键 / 值视图。JSON 是 Java Script Object Notation（Java 脚本对象表示法）的缩写，它是一种描述、序列化及传输数据很方便的方法。它简单易学、易于理解、轻松解析、自我描述、层次分明。另外，JSON 的语法也非常简单。数据由名称 - 值对来表示，并用逗号将其分隔开。对象由大括号将其括起来，数组由方括号将其括起来。

我们常常拿 JSON 与 XML 进行比较，因为它们都是用于数据描述和数据传输的。尽管你会发现 XML 也许是一个更丰富、更具可扩展性的对数据进行序列化和描述的方法，但是你同样也会发现它在读取和解析上的困难性。Hadoop 社区更加喜欢 JSON 而不是 XML。即便如此，在 Hadoop 的基础设施上仍有很多配置文件是用 XML 来编写的，所以为了维持一个 Hadoop 集群，XML 的基础知识仍然是需要的。

3.2.1 教程链接

JSON 已经成为一种数据分享的最受欢迎的标准。所以，在互联网上有非常多的可用信息，如 w3schools 上的文章（*http://bit.ly/1z5nnGr*）。

3.2.2 示例代码

在这里，我们的电影评论数据可以很容易地用 JSON 来表示。

举个例子，这里有原始数据：

```
Kevin,Dune,10
Marshall,Dune,1
Kevin,Casablanca,5
Bob,Blazing Saddles,9
```

在这里，表示成 JSON 格式的数据（该评论被描述为一个称为 `movieReviews` 的集合，它由一个名称 - 值对的集合数组组成，一个是评论家的名字，一个是电影的名字，还有一个是等级评分）为：

```
{
"movieReviews": [
    { "reviewer":"Kevin", "movie":"Dune", "rating","10" },
    { "reviewer":"Marshall", "movie":"Dune", "rating","1" },
    { "reviewer":"Kevin", "movie":"Casablanca", "rating","5" },
    { "reviewer":"Bob", "movie":"Blazing Saddles", "rating","9" }
]
}
```

3.3 Protocol Buffers (protobuf)

许可证	BSD Simplified
活跃度	中
用途	数据序列化
官方网站	https://developers.google.com/protocol-buffers
Hadoop 集成度	API 集成

在这本书中，你会经常看到有关在灵活性和性能之间做权衡这样一个主题。很多时候你只想简单向他人分享数据，这时你将更愿意用牺牲性能来确保数据能很容易地使用。同样，也会有其他一些场合，你需要最大化性能，而为了达到这个目的你将会愿意用牺牲灵活性来换取性能，在这些场合下，你就将需要好好看看 Protocol Buffers 了。

产生这种权衡关系的主要原因是由于 Protocol Buffers 是在编译时组装的。这就意味着当你在构建你的应用程序时，就需要为你的数据进行严格的结构定义，这与 Avro 的运行时组装形成鲜明的对比，Avro 的运行时组装允许你在应用程序运行时定义你的数据结构，或者说 JSON 甚至更加灵活，进行非模式化定义。编译时组装的好处就是那些实际对你的数据进行序列化和反序列化的代码可能会更加优化，当你的应用程序在运行的时候你不需要再为构建这些代码而支付费用。

Protocol Buffers 的目的是快速、简单以及精短。所以，相比与其他的序列化框架（比如 Thrift），Protocol Buffers 支持更少的编程语言和复杂的数据类型。

3.3.1 教程链接

谷歌在官方的项目文档（*http://bit.ly/1yjEVlk*）中提供了多种语言的非常好的教程。

3.3.2 示例代码

与 Avro（在本章前面有详细描述）不一样，Avro 支持运行时 schema 绑定，而 protobuf 必须集成到开发和构建过程中。首先在 *a .proto* 文件里定义一个模型，

比如：

```
message Review {
  required string reviewer = 1;
  required string movieTitle = 2;
  required int32 rating = 3;
}
```

接下来为具体开发语言（比如 Java）运行 protobuf 编译程序，来生成基于模型定义的代码。

通过 protobuf 编译程序针对不同语言的微小变化来生成处理对象的机制。针对 Java 语言，我们使用一个 builder 来创建一个新的、可写的对象：

```
Review.Builder reviewBuilder = Review.newBuilder();
reviewBuilder.setReviewer("Kevin");
reviewBuilder.setMovieTitle("Dune");
reviewBuilder.setRating(10);

Review review = reviewBuilder.build();
```

接下来可以将这个评论对象写入任何形式的输出流：

```
FileOutputStream output = new FileOutputStream("review.dat");
review.writeTo(output);
```

用类型的方式，前面序列化数据的对象重新填充已经完成：

```
FileInputStream input = new FileInputStream("review.dat");
review.parseFrom(input);
```

3.4 Parquet

许可证	Apache License, Version 2.0
活跃度	中
用途	文件格式
官方网站	http://parquet.io
Hadoop 集成度	API 集成

在开放式生态系统的背后，如 Hadoop，一个吸引人眼球的想法是能够为每个具体的任务选择合适的工具。举个例子，为了将数据放到集群上，可以在 distcp（在第 6 章有详细描述）或者 Flume（在第 6 章有详细描述）两个工具之间选择其一；为了构建大数据处理作业，可以在 Java MapReduce 或者 Pig 中选择其一；为管理集群，可以在 Puppet（在第 4 章有详细描述）或者 Chef（在第 4 章有详细描述）中选择其中一个等。这个选择不同于许多传统的平台，它们仅为每个工作提供单一的工具，并且以复杂性作为代价来提供灵活性。

Parquet 是可以选择的众多管理数据存储方式中的其中一个。它是一个柱状的数据存储格式，这就意味着它能很好表现数据的结构化，并且具有相当数量的重复性。在另一方面，Parquet 格式是相当复杂的，当想要在某一时刻检索整个数据记录时它的表现并不是很好。

3.4.1 教程链接

如果你有兴趣学习更多的有关技术工作的内容，GitHub 网站上的 Parquet format 项目（*http://bit.ly/1zWkxHi*）是一个不错的开始。另一个方面，如果你想直接进入实例，可以看 GitHub 网站的 parquet m/r 项目（*http://bit.ly/1KJbOx8*）。

3.4.2 示例代码

许多标准的 Hadoop 工具都支持 Parquet 文件格式，包括 Hive（在第 2 章有详细描述）和 Pig（在第 5 章有详细描述）。通常使用 Parquet 数据格式只需简单地在 CREATE TABLE 命令里添加几行或者在 Pig 脚本上改动几个单词。

举个例子，修改我们的 Hive 实例，用 Parquet 来替代分隔的文本文件格式，我们仅仅在创建表格的时候简单的涉及了 Parquet：

```
CREATE EXTERNAL TABLE movie_reviews
    ( reviewer STRING, title STRING, rating INT)
ROW FORMAT SERDE 'parquet.hive.serde.ParquetHiveSerDe'
STORED
    INPUTFORMAT "parquet.hive.DeprecatedParquetInputFormat"
    OUTPUTFORMAT "parquet.hive.DeprecatedParquetOutputFormat"
    LOCATION '/data/reviews';
```

我们同样也可以修改我们的 Pig 实例，将加载的一个评论文件存储为 Parquet 格式而不是 CSV:

```
reviews = load 'reviews.pqt' using parquet.pig.ParquetLoader
    as (reviewer:chararray, title:chararray, rating:int);
```

第 4 章

管理与监控

构建和密切关注大数据基础设施是一项非常艰巨的任务。各种各样的软件分布在各种不同的机器上，而这些机器又有着截然不同的配置。如果系统的一个部件失败了将如何得知，在解决这个问题之后，又将如何对这部分组件进行备份呢？如何能够让系统中各个不同的部件之间进行通信，让它们能够和许多活动部件一起完成艰巨的工作呢？

幸运的是，大数据生态系统提供了很多工具，可以帮助你减轻管理和监控架构的负担。我们主要介绍三种主要的工具：

节点配置管理

诸如 Puppet 或者 Chef 这样的工具可以帮助用户管理配置系统。它们能够做一些如改变操作系统参数、安装软件这样的工作。

资源跟踪

尽管在架构上会有许多随工具而来的独立的组件，它们可以帮助监视特定组件的性能，但是有时还是需要一个独立的观测仪表板，且它们不依赖于任何特定工具而存在。

协调

很多任务使用了大数据系统中各种的不同的组件。像 ZooKeeper 这样的工具就可以帮助用户对所有这些活动部件进行同步，以此来完成共同的目标。

4.1 Ambari

许可证	Apache License, Version 2.0
活跃度	高
用途	配置、监视和管理 Hadoop 集群
官方网站	http://ambari.apache.org
Hadoop 集成度	完全集成

如果你曾经试图从 Apache 上下载并安装 Hadoop，你就会知道 Hadoop 的安装和管理仍然是非常困难的。近来，两个主要的供应商 Pivotal 和 Hortonworks 在 Ambari 上展开合作，尝试生产基于 RESTful API 的生产就绪的、易于使用的、基于 Web 的 GUI 工具。

Ambari 官方网站的文档中说它可以实现如下功能：

- 提供和监控 Hadoop 集群。
- 在任意数量的主机上通过一步一步地安装向导来安装 Hadoop 服务。
- 在整个集群上为 Hadoop 服务器提供启动，停止以及重新配置的集中管理。
- 为监控 Hadoop 集群的健康度和状态提供了可视化界面。
- 使用 Ganglia（在本章后面有详细描述）收集度量标准。
- 用 Nagios（在本章后面有详细描述）支持系统报警。

尽管使用传统的方法来安装 Hadoop 可能是一个不止一日的折磨过程，但是 Ambari 却能在几个小时内轻松完成这些工作。2013 年末 Ambari 从孵化器毕业，成为顶级项目，现在应该已经做好生产使用的准备了。

4.1.1 教程链接

有一个无声视频（*http://youtu.be/xeT3A1nha6g*）可以引导你通过 Ambari 来构建集群。

这里也有一个非常好的幻灯片教程（http://slidesha.re/16SvC1L）。

4.1.2 示例代码

Ambari 是一个基于 GUI 的工具，所以我们没有办法提供一个代码的例子。

4.2 HCatalog

HCatalog
Table Management

许可证	Apache License, Version 2.0
活跃度	高
用途	数据抽象层
官方网站	http://hive.apache.org/javadocs/hcat-r0.5.0/index.html
Hadoop 集成度	完全集成

假设要通过 MapReduce、Hive 和 Pig 来访问你一系列的文件。如果有一种可以访问到它们，而你不需要知道文件格式和位置这样一些细节的方法是不是非常有用呢？你肯定会表示同意。HCatalog 为很多在 HDFS 上的文件类型提供了一个抽象层，它允许用户的 Pig、Hive 和 MapReduce 可以将注意力放在读写数据上，而不用去详细考虑到底使用了哪个格式。这个抽象层使数据看起来很像关系数据（也就是，使用由行、列组成的表格来组织，十分类似 SQL 的感觉）。HCatalog 与 Hive 密切相关，因为它利用了 Hive 的元存储，且在这里 Hive 存储了关于其表的元数据。

HCatalog 有个分区的概念。一个分区是一个表上行数据的子集，它们拥有共同的特性。通常情况下，表按时间字段来分区。这样方便查询也容易管理，当这些分区不再需要的时候可以将它们删除。

如果决定使用 HCatalog，你就可以通过 HCatalog 方法来访问你的数据，而不是使用原始的 Pig 或 MapReduce 来访问你的数据了。举个例子，在 Pig 里，你通常使用 PigStorage 或者 TextLoader 来读取数据，而当使用 HCatalog 后，你可以使用 HCatLoader 和 HCatStorer。

4.2.1 教程链接

HCatalog 是 Hadoop 生态系统中文档较少的主要项目之一，但是这个来自 HortonWorks 的教程（*http://youtu.be/_dVlNu4lqp*E）还是很好的。

4.2.2 示例代码

仅仅通过 Pig 而不使用 HCatalog，你可以通过使用类似下面的代码来加载文件：

```
reviews = load 'reviews.csv' using PigStorage(',')
    as (reviewer:chararray, title:chararray,rating:int);
```

如果要使用 HCatalog，你可能首先要在 Hive 里创建一个表：

```
CREATE TABLE movie_reviews
    ( reviewer STRING, title STRING, rating INT)
ROW FORMAT DELIMITED
FiELDS TERMINATED BY '|'
STORED AS TEXTFiLE
```

然后在 Pig 语句中使用它：

```
reviews = load 'movie_reviews'
USING org.apache.hcatalog.pig.HCatLoader(); -
```

4.3 Nagios

许可证	GNU General Public License
活跃度	高
用途	IT 基础设施监控
官方网站	http://www.nagios.org
Hadoop 集成度	不集成

那些曾经负责管理网络计算机系统的人们知道，跟踪并记录那些在网络里发生的事情是非常重要的。你需要在有事情出错的时候被警告而不是通过手动查询来发现错误。你更希望能够自动地重启那些失败的组件。同样你更希望有一个工具可以提供一个图形化界面，这样你可以快速地查看在环境中所发生的事情。Nagios就是这样一个工具，和许多开源的产品一样，它有一个可供你免费下载的版本，而扩展版本的许可证是要需要付费的。

开源的核心产品有许多有用的特性。它通过基于 Web 的界面提供了对服务器、交换机、操作系统以及应用程序监控。更重要的是，它提供了对停电及故障的快速检测，且通过email或者文字信息向你的操作人员发出警告。它还可以自动重启。

更重要的是，Nagios 可以嵌入到其他的系统中，包括 Ambari（在本章前面有详细描述）。

4.3.1 教程链接

Nagios 主页上有一个生动演示系统（*http://bit.ly/1Chq6yn*）。

这里也有一些其他的资料，如 Debian 帮助（*http://bit.ly/1zLS1JC*）上的内容以及 TuxRadar（*http://bit.ly/1CgXTuc*）上的内容。

4.4 Puppet

许可证	Apache License, Version 2.0
活跃度	高
用途	节点管理
官方网站	https://puppetlabs.com
Hadoop 集成度	API 兼容

Puppet 是一个很受欢迎的管理大量机器配置的系统。它使用一个“声明式”（declarative）语法，这就意味着用户要对机器配置进行描述，然后由 Puppet 来负责搞清楚要实现该配置所必需的步骤。举个例子，你先描述一个特定程序安装的配置，然后 Puppet 将负责计算出如果这些程序已经安装它将如何停止，以及如果还没有安装将如何去安装。

Puppet 的配置按照 resource、manifest 和 module 来编写。resource 是配置中最基本的单元，它表示特定事情的状态。举个例子，一个 resource 是一个特定文件可能存在的状态，并且每一个人都应该可以查看到它的内容，但是仅仅只有系统管理员可以改变它的内容。

配置的下一个层次就是 manifest。一个 manifest 是一组相关的 resource。举个例子，一个 manifest 可能会提示安装你的应用程序同时需要一个特定版本的 RedHat 软件包管理工具（RedHat Package Manager，RPM）被安装，以及该配置描述被创建。

一个 module 是一组逻辑相关 manifest，但它们之间是相互独立的。举个例子，我们应用程序的一个 module 可能包括：一个用来安装应用程序的 manifest，一个用来配置应用程序使其能使用 HBase 实例（在第 2 章有详细描述）的 manifest，以及一个用来配置应用程序使其能将日志发送到特定的位置上的 manifest。

4.4.1 教程链接

Puppe 实验室为初学者提供了许多资源（*http://bit.ly/17gLBYA*）。

4.4.2 示例代码

Puppe 的 manifest 是用 Ruby 来编写的，它遵循 Ruby 语法的规则。

这里 manifest 的例子是安装我们应用程序，确保配置目录存在的写法，类似如下所示：

```
# 'test_application.pp'
classtest_application {
    package { 'test_application':
        ensure => installed
    }

    file { 'test_application_conf':
        path => '/etc/test_application/conf',
        ensure => directory,
        require => Package['test_application']
    }
}
```

4.5 Chef

许可证	Apache License, Version 2.0
活跃度	高
用途	节点管理
官方网站	https://www.getchef.com
Hadoop 集成度	API 兼容

设计 Chef 的目的是为了减轻管理配置基础设施的负担。它遵循一个很多软件开发者都很熟悉“命令式”（imperative）语法，该语法允许它们可以像编写软件代码一样编写软件配置。

Chef 配置编写为 resource、recipe 和 cookbook。resource 是配置中最基本的单元，它描述了如何配置一个特定的事情。举个例子，一个 resource 可以告诉 Chef 去创建一个特定的目录，并确保每个人可以看到它的内容，但是只有系统管理员可以改变它的内容。

配置的下一个层就是 recipe。一个 recipe 是一组相关的 resource。举个例子，一个 recipe 可能会告诉你安装你的应用程序需要两个步骤：首先你必须安装一个特定的包，接下来你必须创建一个配置目录。

一个 cookbook 是一组逻辑相关的 recipe，但它们之间是相互独立的。举个例子，我们应用程序的一个 Chef 可能包括：一个用来安装应用程序的 recipe，一个用来配置应用程序使其能使用 HBase 实例的 recipe，以及一个用来配置应用程序使其能使用 Accumulo 实例的 recipe。以上这三个 manifest 涉及安装，配置我们的应用程序，但是需要按实际情况来控制这些 manifest 的运行，而不是要求它们所有都运行。

4.5.1 教程链接

Opscode 在它的维基百科页面（*https://wiki.opscode.com*）上提供了大量的资料，这是一个很好的开始。

4.5.2 示例代码

Chef 的 recipe 由 Ruby 编写，遵循典型的 Rudy 语法规则。

这里有一个 manifest 的例子，用来安装应用程序，并且确保配置目录的存在：

```
# 'default.rb'
package "test_application" do
    action :install
end

directory "/etc/test_application/conf" do
    action :create
end
```

4.6 ZooKeeper

许可证	Apache License, Version 2.0
活跃度	中
用途	协调
官方网站	https://zookeeper.apache.org
Hadoop 集成度	API 兼容

Hadoop 和 HDFS 是跨机器分配工作最有效的工具，但是有时候你需要快速地在同时运行的众多进程中分享很少的一点点信息。ZooKeeper 的建立正是出于这样的一个需要：它是一个跨机器存储和分享少量状态以及配置数据的有效机制。

举个例子，你有一个任务，要从大量的小文件中获取信息，将它们转化成数据，并将这些信息放在数据库上。

你可以将这些信息存放在一个共享文件中或 HDFS 上，但是从众多机器上访问这些信息会非常慢，且尝试对信息的更新会非常困难，因为同步有可能会引起问题的产生。

稍微好一点的方法就是将关联信息也移动到一个 MapReduce 工作配置文件上。但即使那样，你还是需要为每一次的分析，每一次的数据库移动而进行文件的更新。同样，如果当你有一个分析正在运行时，需要移动数据库，是没有更简单的方法来更新关联信息的。

更好一点做法是，将关联信息存放在 ZooKeeper 上，它允许你在快速分析访问信息的同时也提供了一个简单的更新机制。

ZooKeeper 并不打算填补 HBase（在第 2 章有详细描述）或任何其他大数据键 - 值存储的空缺。实际上，ZooKeeper 中保护措施的建立是为了确保人们不要尝试

将它作为一个大数据存储来使用。然而，ZooKeeper 仅仅是在当你想要跨越你的环境分享少量信息时使用。

4.6.1 教程链接

官方的入门指南（*http://bit.ly/1DFmPNU*）是一个你开始尝试 ZooKeeper 的很好的地方。

4.6.2 示例代码

在这个例子中，我们首先要打开 ZooKeeper 的命令行界面：

```
$ zookeeper-client
```

现在我们来创建一个键 - 值对，在这个例子中，关键字是 `/movie_reviews/database`，值是我们用来存放电影评论的数据库的 IP 地址：

```
[zk: localhost:3000(CONNECTED) 0] create /movie_reviews ''
Created /movie_reviews

[zk: localhost:3000(CONNECTED) 1] create /movie_reviews/database
'10.2.1.1'
Created /movie_reviews/database
```

现在可以获取关键字的值。请注意，我们得到了两个重要的数据块：10.2.1.1 的实际的值以及该值的版本：

```
[zk: localhost:3000(CONNECTED) 2] get /movie_reviews/database
'10.2.1.1'
<metadata>
dataVersion = 0
<metadata>
```

想象一下，托管数据库的原始服务器崩溃了，需要为故障转移提供所有使用了该数据库的不同的进程。我们更新了与关键字相关联的值，指向我们故障转移的 IP 地址：

```
[zk: localhost:3000(CONNECTED) 0] set /movie_reviews/database
'10.2.1.2'
<metadata>
dataVersion = 1
<metadata>
```

现在让我们最后一次获取关键字。请注意，我们重新获取了新的IP地址以及已经增加的版本号，这说明值已经改变了：

```
[zk: localhost:3000(CONNECTED) 1] get /movie_reviews/database
'10.2.1.2'
<metadata>
dataVersion = 1
<metadata>
```

4.7 Oozie

许可证	Apache License, Version 2.0
活跃度	高
用途	一个管理复杂的 Hadoop 多部件工作的工作流调度
官方网站	https://oozie.apache.org
Hadoop 集成度	完全集成

你可以通过一个独立的 MapReduce、Pig 或者 Hive 工作来完成数据的分析工作，它们从分布式文件系统（HDFS，在第 1 章有详细描述）上读取数据，对这些数据进行计算然后将它们输出存放在 HDFS 上，但是这一系列的任务将会非常复杂。举个例子，现在有一个工作，它需要建立在其他两个或三个工作完成的基础之上，而每一个所需的数据都要从一些外部的资源加载到 HDFS 上。并且你希望这些工作能定期地运行。当然，你完全可以安排通过手动来实现或者通过一些较好的脚本来实现，但是这里有一个更为简单的方法。

这个方法就是 Oozie，一个 Hadoop 的工作流调度程序。在刚刚开始的时候，它有一点点复杂，但是它对启动、停止、暂停、重启这样一些工作以及控制整个工作流程都是非常有用，所以当需要的任务和对象还没有准备好之前在完整的工作中是没有作业运行的。Oozie 将它的 action（工作和任务）放在一个有向非循环图（DAG）中来描述那些需要依赖前序 action 成功完成的 action。这是在一个非常大的 XML 文件（实际上是 hPDL，Hadoop 的流程定义语言）中定义的。这个文件太大了，我们没有办法在这里通过例子来展示，但是教程和 Oozie 网站上有实例介绍。

什么是 DAG 呢？它是一个由节点和弧的集合组成的图。节点代表着声明（state）或对象。弧将节点连接起来。如果一个弧的一端有箭头，那么这个弧就表示是一个控制弧，它的方向就是箭头指向的方向。在 Oozie 中，节点就是 action，比如运行的工作，分支，失败或者结束。弧表示 action 的流向。它直接显示了 action 的先后次序，决定和控制哪些节点工作必须在某些工作之前或者之后运行（比如，在一个 Pig 脚本运行之前一个文件对象必须存在）。非循环意味着在遍历图中一

旦你离开某个节点，你将不可能再次回到这里了。这就是一个周期。这也就意味着 Oozie 不可能通过在一组节点上进行迭代直到达到某个条件（比如，这里没有 while 循环）。在“Giraph”（在第 2 章有详细描述）上有更多的关于图的信息。

图 4-1 是一个 Oozie 流程的图形的例子，在这个图中要开始一个 Hive 工作的要求是，输入一个 Pig 作业和一个 MapReduce 作业。这两个作业都需要外部文件。

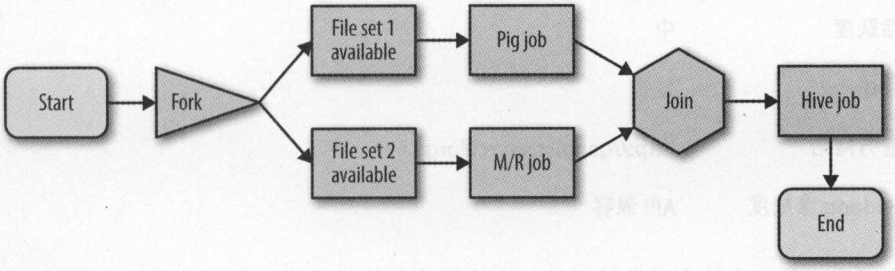


图 4-1：Oozie 工作流的图形表示

Oozie 的安装提供了一个可以完成监控工作的 GUI 控制台，但是需要使用 Ext JS（这是一个构建桌面应用程序的 JavaScript 框架），它同样提供了开源程序和商业版本两个版本。

最近已经有一些地方利用 Hue 来作为更通用的 Hadoop 的监测工具。Hue 是一个开源的工具，发布在 ApacheLicense，Version 2.0 下，主要与 Cloudera Hadoop Distribution 相关。

4.7.1 教程链接

这里有一些有趣的 Oozie 演示，如 IBM 的大数据团队做的工作（*http://youtu.be/ypn7dGkD3mA*）以及学术演讲（*http://youtu.be/GGRhSEE45tk*）。

4.7.2 示例代码

实际的示例文件太大了，不太适合放在这里。更多的信息可以参考官方的用户资源页面（*http://bit.ly/1EYrsnK*），它包括各种语言的经典实例。

4.8 Ganglia

许可证	BSD
活跃度	中
用途	监视
官方网站	http://ganglia.sourceforge.net
Hadoop 集成度	API 兼容

Ganglia 是一个分布式监控系统，它设计的目的就是能与众多机器组成的集群和网格一起工作。它可以将你系统的使用快速可视化，且是跟踪集群福利的一个非常有用的工具。Ganglia 最适合在一个广而浅的层次上对系统行为进行理解。那些正在寻找调试或优化特定分析的人们也许会更加看好其他的那些面向在更窄的范围内提供更深层次的信息的工具，比如 White Elephant。

默认情况下，Ganglia 能够提供很多关于系统内部的工作原理信息。这包括一些数据点的描述，诸如能够使用的总的计算能力、通过网络移动的数据大小，以及能够持久利用的存储容量等。如果用户有额外的需求，同样可以通过使用一些插件来扩展 Ganglia 从而捕获和显示到更多的信息，如应用程序特定的指标。Hadoop 与一系列插件一起打包用来向 Ganglia 报告有关 HDFS 和 MapReduce 的信息（查看 Ganglia Metrics“*http://bit.ly/1ITRL1r*”项目获取更多信息）。

Ganglia 在 Ambari（在第 4 章有详细描述）项目中被使用。

4.8.1 教程链接

Ganglia 有一个广泛的分布的网络支持（*http://bit.ly/1CgY34H*），包括邮件列表、GitHub 缺陷跟踪、维基百科页面等。当人们开始他们的第一个 Ganglia 安装工作的时候，从维基百科页面（*http://bit.ly/1vkfVXp*）开始是一个不错的选择。

4.8.2 示例代码

配置一个 Ganglia 实例已经超过了本书的范围，因为这是一个最基本的分布式系统监测和诊断的过程。鼓励有兴趣的读者去查看监控维基百科 (Wikipedia) 集群（*http://bit.ly/1FyOQpx*）的 Ganglia 实例，在实践中理解 Ganglia。

第 5 章

分析辅助

当前阶段我们已经可以将数据放入 Hadoop 集群，那么接下来该做些什么呢？通常用户希望从简单的数据净化或数据变换开始。该操作可能是简单地重组字段和移除破损的记录，也可能涉及复杂的各种形式的聚合、浓缩和归纳。一旦完成对数据的清理后，用户可能会满足于简单地将其放入一个更为传统的数据存储中，例如关系型数据库，然后开始考虑需要做的大数据工作。另一方面，你可能会希望继续处理数据，通过运行专门的机器学习算法来对数据进行分类，或者执行一些基于地理空间的排序分析。

在本章中，我们将开始讨论两种类型的工具：

Mapreduce 接口

这是帮助用户更方便处理数据的通用工具。

分析库

这是一个专用库，包含了一些能简化数据分析的功能。

5.1 MapReduce 接口

在早期的 Hadoop 中，处理系统的数据只有一种方法，就是使用 Java 语言开发 MapReduce 程序，但是该方法存在若干问题：

- 开发者在分析程序的时候不仅必须理解业务逻辑和数据，还需要理解 Java 代码。
- 将一个 Java 归档传输到 Hadoop 中，比简单地编写一个查询语句更耗时。

例如，对一个开发者来说，直接通过 MapReduce 开发和测试一个简单的分析程序的步骤如下所示：

1. 编写几百行用 Java 语言编写的 MapReduce 代码。
2. 将代码编译为 JAR 文件（Java 归档）。
3. 将 JAR 文件拷贝到集群中。
4. 运行分析器。
5. 查找 bug，然后回到前面步骤编写更多的代码。

正如你所想象的，该过程十分消耗时间，并且胡乱地对代码进行修修补补，可能会中断对业务逻辑问题的思考。幸运的是，现在出现了一个强健的生态系统工具，该工具可以和 Hadoop 和 MapReduce 一起工作，并简化处理的进程，这使得分析员可以集中更多时间来思考手头上的业务逻辑问题。

如你所见，这些工具通常做如下事情：

- 提供一个更简单更熟悉的 MapReduce 接口。
- 通过支持用户构建交互式查询来提供即时反馈。
- 简化复杂操作。

5.2 分析库

尽管在 MapReduce 或 Pig 中就可以进行多数的分析，但是，现在还有很多机器学习的算法是作为 Apache Mahout 项目的一部分来发布的。例如采用 Mahout 比较适合解决分类、推荐和聚类等算法的问题。

一旦将机器学习算法指向一个数据集，接下来这些算法就可以自动从数据中开展学习了。这些算法可以分为两类：有监督的和无监督的。在有监督的学习中，通常数据有一个用于观测的数据集和一个输出值。举例来说，关于病人的临床数据是可以观测的，而输出值就是是否存在疾病。在一个有监督的学习算法中，给定一个新病人的临床数据，尝试去预测某种疾病是否存在。无监督的算法不使用一个给定的输出，取而代之的是去尝试寻找数据中一些隐藏的模式。举例来说，我们可以从病人中获取可观测的临床数据集，并尝试查看这些数据是否易于聚类，

以便在簇中的点距离簇中心比其他簇中心要近。簇的解释不是由算法给出的，而是要数据分析者来发现的。可以在 Mahout 的主页（*http://bit.ly/1vkgb8Q*）中查到支持的算法列表。

推荐算法的基本思路如下：基于其他人员的评价，以及他们与你的相似度，来判断你最可能对哪些给出更高的评价。

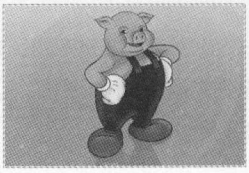
分类算法是指给定关于某个个体的观测集，预测未知的输出。如果输出是一个二态变量，可以使用逻辑回归来预测输出的可能性。例如，给出一名病人的实验结果，预测该病人得某种疾病的概率。如果输出一个数值变量，可以使用线性回归来预测输出的值。举例来说，给出月度经济状况，可以预测下一个月的失业率。

聚类算法不需要真正回答一个问题。可以在分析的初始阶段频繁使用该算法，以便获取对数据的基本感觉。

数据分析是一个很有深度的话题，这里没法详细讨论。O'Reilly 出版社出版了一系列非常优秀的关于数据分析的书籍（*http://oreil.ly/1ITSeka*）。

大多数的分析仅仅讨论如何处理数值型或分类数据。在 Hadoop 的世界中文本分析和地理空间分析将更为重要。

5.3 Pig



许可证	Apache License, Version 2.0
活跃度	高
用途	处理数据的高级别数据流语言
官方网址	http://pig.apache.org
Hadoop 集成度	完全集成

如果我们把 MapReduce 称为 Hadoop 中的“汇编语言”，那么 Pig 则可以比作 Python 或其他高级语言。为什么用户更希望使用 Pig 而不是 MapReduce 呢？虽然使用 Pig 语言编写的程序可能在性能方面没有使用 Java 编写的 Map 和 Reduce 函数的性能高，但是使用Pig可以提高开发者的编码速度，并使得代码维护更方便。Pig 自称为一门数据流语言，在使用该语言时，可以使用一种类似 SQL 语言结构的程序组合来读取和变换数据集。

Pig 语言的名字来自“猪吃任何东西”，意味着 Pig 语言可以容纳多种不同格式的输入，但通常用于变换文本数据集。在大多数情形下，Pig 是一个极好的 ETL（抽取/转换/加载）工具。Pig 被翻译为或编译为 MapReduce 代码，并被合理地优化（使得并不是一系列 Pig 语句中的每个语句都生成 Map 函数和 Reduce 函数），然后顺序执行。

在 Piggy Bank（*http://bit.ly/1EYrPyH*）中有一个共享的 Pig 程序库。

5.3.1 教程链接

“*http://bit.ly/199tCnF*”提供了一个十分完整的向导来全程指导用户安装 Pig，并编写了最初的几个程序脚本。“working with Pig”（*http://youtu.be/tFsHO12eOgc*）是 Pig 技术的一个很好的概览。

5.3.2 示例代码

在 Pig 中，只需要 5 行代码就可以快速地描述电影评论问题。

```
-- 读取所有电影评论，并查找平均评价
--reviews.csv 文件中各行格式如下：
  name, film_title, rating
  reviews = load 'reviews.csv' using PigStorage(',')
   as (reviewer:chararray, title:chararray,rating:int);

-- 只考虑关于 Dune 的评论
   duneonly = filter reviews by title == 'Dune';

-- 希望使用 Pig 内置的 AVG 函数，但是 AVG 只适用于 bag，而不是适用于 list 类型，因
此这里创 -- 建 bag 类型
   dunebag = group duneonly by title;

-- 现在开始生成平均值，并转储文件
   dunescore = foreach dunebag generate AVG(dune.rating);
   dump dunescore;
```

5.4 Hadoop Streaming

许可证	Apache License, Version 2.0
活跃度	中
用途	使用 Java 外的其他语言来编写 MapReduce 代码
官方网站	http://hadoop.apache.org/docs/r1.2.1/steaming.html
Hadoop 集成度	完全集成

如果有一些数据，并且你对如何处理这些数据有一些想法，你理解 MapReduce 概念的思想，但是你在 Java 和 MapReduce 方面的经验并不丰富，且该问题又不适用其他 Hadoop 提供的主要的工具。那么 Hadoop Streaming 可能是一个解决方案，它允许开发者使用各种支持 Linux 系统标准输入读取和标准输出写入特性的语言。

开发者仍然需要编写 Map 和 Reduce 函数，但是可以选择自己喜欢的开发语言。Map 函数可以从一个文本文件中读取数据，并生成由 tab 字符隔开的键值对。处理的 Shuffle 阶段将由 MapReduce 基础架构来处理，而开发者的 Reduce 函数将从标准输入（stdin）中读取数据，进行处理，然后输出到标准输出（stdout）中。

下面“教程链接”一节中将会展示使用 Python 语言通过 Hadoop Streaming 开发的一个 WordCount 应用程序。

Hadoop Streaming 的性能能做到和原生的 Java 代码一样吗？几乎可以肯定地说性能不如原生的 Java 代码，但是如果你的组织拥有 Ruby 或 Python 类似的开发能力，对于你来说这是更好的选择，而不是让开发者在开发 MapReduce 项目前，立刻去学习 Java 语言。

5.4.1 教程链接

在“*http://bit.ly/1DhVReH*”网址有一些很优秀的关于本技术的概论，同时还提供了一个教程。

5.4.2 示例代码

现在将使用 Hadoop Streaming 来计算关于 *dune* 的平均评价。首先使用下面一个小的数据集：

```
Kevin,Dune,10
Marshall,Dune,1
Kevin,Casablanca,5
Bob,Blazing Saddles,9
```

那么 Map 函数如下所示：

```
#! /usr/bin/python

import sys

for line in sys.stdin:
    line = line.strip()
    keys = line.split(',')
    print( "%s\t%s" % (keys[1], keys[2]) )
```

Reduce 函数如下所示：

```
#!/usr/bin/python

import sys

count = 0
rating_sum = 0
for input_line in sys.stdin:
    input_line = input_line.strip()
    title, rating = input_line.split("\t", 1)
    rating = float(rating)
    if title == 'Dune':
        count += 1
        rating_sum += rating
dune_avg = rating_sum/count
print("%s\t%f" % ('Dune',dune_avg))
```

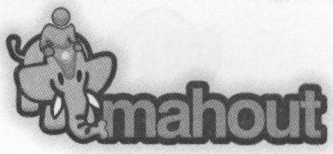
该任务运行结果如下：

```
hadoop jar contrib/streaming/hadoop-*streaming*.jar \
-file /home/hduser/mapper.py \
-mapper /home/hduser/movie-mapper.py \
-file /home/hduser/reducer.py \
-reducer /home/hduser/movie-reducer.py \
-input /user/hduser/movie-reviews-in/* \
-output /user/hduser/movie-reviews-out
```

处理结果为：

```
Dune 5.500000
```

5.5 Mahout



许可证	Apache License, Version 2.0
活跃度	高
用途	机器学习和数据分析
官方网站	http://mahout.apache.org
Hadoop 集成度	API 兼容

如果在 Hadoop 中有许多数据，你将如何处理它们呢？你可能希望做一些分析，或者数据科学，或者是机器学习。大多数工作可以使用标准 Apache 发行版本中提供的工具，例如 Pig、MapRecue 或 Hive。但是如果有更复杂的应用，将需要使用一些算法，而你可能并不希望自己编写代码来实现这些算法。因此你可以求助于 Mahout。什么是 Mahout？ Mahout 是一个支持在 Hadoop 上运行的可扩展机器学习算法的集合。为什么给它起名为 Mahout？ Mahout 是北印度的管象人的名字，正如 logo 中所看到的那样。下面列出的算法正在持续增长，截至目前，主要包括如表 5-1 中所示的算法。

表 5-1：Mahout MapReduce 算法

Mahout 算法	简要描述
K 均值 / 模糊 k 均值聚类	聚类是将一个可观测的集合划分为多个组，组内的元素都相近，而组与组之间的元素有差异
潜在狄利克雷分布（Latent Dirichlet allocation）	LDA 是一种建模技术，该技术是一种常用的文档分类技术，该技术基于文档中特定主题来对文档进行分类
特征值分解（Singular value decomposition）	如果没有许多线性代数和特征值的背景，还是很难清楚地解释 SVD
基于逻辑回归的分类器在	逻辑回归通常用于预测值为 0~1 的变量，例如是否存疾病、是否具有组织中成员资格等
补充朴素贝叶斯分类器（Complementary naïve Bayes classifier）	这是另外一个利用贝叶斯定理（可能你还记得在统计学中学习过）的分类器方案

表 5-1：Mahout MapReduce 算法（续）

Mahout 算法	简要描述
基于随机森林决策树分类器	也是另外一个基于决策树的分类器
协同过滤	用于推荐系统中（如果用户喜欢 X，那么系统为用户推荐 Y）

对上述技术进行全面深入的讨论已经超出了本书的范围。现在已有许多关于机器学习很好的介绍。Google 将会是你的好朋友。

在 2014 年 4 月，Mahout 社区宣布正在将开发语言由基于 MapReduce 改为一个基于 Scala 的专用领域语言，Scala 是 Spark 的具体实现（详见第 1 章描述）。现有的 MapReduce 算法仍然受到支持，但是除此之外的其他一些代码并不是基于 MapReduce 的，实际上，Mahout 社区已经放弃了对一些不常用的程序的支持。

5.5.1 教程链接

Mahout 用户有一个完整的关于书籍、指南和演讲的策划链接地址是“http://bit.ly/1zLSJqu”。

5.5.2 示例代码

使用 Mahout 来处理推荐系统的进程有些过于复杂，在这里就不详细描述了。Mahout 包含一个电影评分推荐系统的范例（*http://bit.ly/1zLSKKM*）。可以通过 GroupLens Research（*http://bit.ly/1vEPilR*）获取数据。

5.6 MLLib

许可证	Apache License, Version 2.0
活跃度	高
用途	Spark 的机器学习工具
官方网站	https://spark.apache.org/mllib
Hadoop 集成度	完全集成

如果你决定投入到 Spark 中，且需要一些机器学习的工具，那么 MLLib 给你提供了一个基本的集合。类似 Mahout 中提供的功能性（详见本章前面描述），MLLib 有一个增长的模块列表，这些模块对于数据科学家和大数据分析团队执行任务是很有用的。在 1.2 版本中包含了表 5-2 列出了一些模块（包含但不限于这些模块）。新的算法也正在不断增加。

表 5-2：MLLIb 算法

MLLib 算法	简要描述
线性 SVM 和逻辑回归	使用连续和二元变量来进行预测
分类和退化树	基于二元判定的分类数据方法
K 均值聚类	聚类是将一个可观测的集合划分为多个组，组内的元素都相近的，而组与组之间的元素有差异
使用交替最小二乘法的推荐系统	用于推荐系统中（如果用户喜欢 X，那么系统为用户推荐 Y）
多项朴素贝叶斯	基于贝叶斯理论的分类器
基本统计算法	概要统计、随机数据生成、相关性
特征提取与转换	通常用于文本统计的多种常用方法
降维	减少分析问题中的变量数量，通常用于高度相关的时候

因为 MLLib 是基于 Spark 的，所以你最好对 Scala、Python 或 Java 有一定了解，以便需要的时候使用这些语言来做一些复杂的处理。

你可能想知道到底是该选择 MLLib 还是 Mahout。从短期来看，Mahout 更为成熟，并已经有更大的范例集合，但是当前的 Mahout 版本使用 MapReduce，通常来说，MapReduce 比 Spark 更慢（尽管 MapReduce 可能更稳定）。如果你需要的算法现在只在 Mahout 上面提供，那么你就不会为选择哪个平台而困扰了。Mahout 的社区现在用户数更多，因此如果你寻求在线帮助，你可能更容易找到 Mahout 的解决方案。另一方面，Mahout v2 版本将迁移到 Spark 和 Scala 上，因此长久来看，MLLib 将可能代替 Mahout，或者这两者将合并。

5.6.1 教程链接

“MLLib: Scalable Machine Learning on Spark”（*http://stanford.io/1E9CGlF*）是一个很完整，有一定技术性的教程，用户可能会发现这十分有用。

5.6.2 示例代码

伯克利大学的 AMPlab 实验室提供了一些范例代码（*http://bit.ly/1CgYmMV*），该范例基于协同过滤来进行个性化的电影推荐。

5.7 Hadoop 图像处理接口（HIPI）

许可证	BSD Simplified
活跃度	中等
用途	图像处理
官方网站	http://hipi.cs.virginia.edu/index.html
Hadoop 集成度	API 兼容

图像处理是一个被广泛重载了的术语。它可以指通过简单地将图片放入焦点，来使得边界清晰，进而清理图片。也可以指判定图像中有什么，进行场景分析。举例来说，肺部的 X 射线照片是否显示有肿瘤？子宫颈抹片检查采集的细胞图片是否表明有潜在的宫颈癌？某个指纹图像是否匹配特定图像？或者与某一系列图像类似？

HIPI 是 Virginia 大学开发的一个图像处理程序包。相关文档十分概略，该程序包主要用于对采集的图像进行检查，并对这些图像的相似度和非相似度进行判定。该程序包假定用户了解图像处理领域的专用技术，例如了解可交换图像文件（exchangeable image file format，EXIF）。因为这是一个大学的研究项目，所以该项目的未来有不确定因素，但是看起来在 2015 年该项目的活跃度有所复活，包括 Spark 的整合。

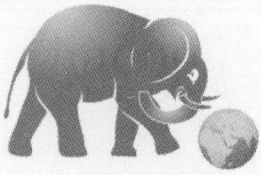
5.7.1 教程链接

HIPI 仍然是一个十分新的技术，当前最好的信息源来自于这篇博士论文（*http://bit.ly/1FyPjYS*）。

5.7.2 示例代码

可以在该项目的官网（*http://bit.ly/1zWlFe4*）上找到许多 HIPI 使用的范例。

5.8 SpatialHadoop



许可证	未知
活跃度	高
用途	空间分析
官方网站	http://spatialhadoop.cs.umn.edu
Hadoop 集成度	API 兼容

如果你过去做过大量与空间数据相关的工作，那么你可能对 PostGIS 比较熟悉，这是一个开源 PostgreSQL 数据库上的开源空间扩展程序。但是如果你希望在大规模的 Hadoop 环境中应用，而不是在 PostgreSQL 中，该如何办呢？ Minnesota 大学的计算机系开发了 SpatialHadoop，这是一个 MapReduce 的开源扩展，设计用于处理 Hadoop 中海量的空间数据集。要使用 SpatialHadoop，首先需要将数据加载到 HDFS 中，然后构建一个空间索引。一旦完成对数据的索引后，就可以执行 SpatialHadoop 提供的各种空间操作，例如范围查询、k 近邻和空间 join 等操作。

由于 SHadoop 提供了高级别的调用，可以生成 MapReduce 任务，所以使用 SHadoop 时可以不需要编写 MapReduce 代码。网站上提供了清晰的使用示例。

除了有基于 MapReduce 的实现以外，现在也有基于 Pig 的扩展，名为 Pigeon，该扩展支持用户使用 Pig Latin 语言来进行空间查询。可以从项目的 GitHub 主页（*http://bit.ly/1zWlJdY*）获取 Pigeon。Pigeon 声称目标是支持尽可能多的 PostGIS 功能。这是一个有雄心并十分有用的目标，因为 PostGIS 有很广泛的追随者，并且它所支持的 ST 功能使得空间分析十分简单，可以使用类似 Pig/Pigeon 这样的高级语言来实现。

可以从 GitHub 的相关网址（*http://bit.ly/1ITT0gY*）中获取开源代码。

5.8.1 教程链接

SpatialHadoop 项目的官方网站提供了大量很好的教程（*http://bit.ly/1Md6lQQ*）。

第6章

数据传输

数据传输主要需处理3个关键问题：

- 怎样将数据放到Hadoop集群上？
- 怎样从Hadoop集群上获取数据？
- 怎样将数据从一个Hadoop集群上移动到另一个Hadoop集群上？

一般来说，Hadoop不是一个事务引擎，事务引擎加载的数据信息量少，内容离散且关联信息少，就像在航空售票系统里的那样。相反，从外部来的数据是批量加载的，比如加载来自外部传感器的平面文件，批量加载类似美国联邦政府网（*http://www.data.gov*）上的数据或日志文件，或从关系系统上转移数据。

Hadoop生态系统包含了各种各样的非常好的工具，它们可以对你的数据进行很好处理。然而，以Hadoop开始或结束你的数据都是非常少见的。较为常见的是有一个工作流，该工作流起始于那些来自外系统的数据，比如来自Web服务器上的日志，并在托管于一个商业智能（BI）系统上结束分析。

数据传输工具帮助数据在这个系统之间进行移动。更进一步明确地说，数据传输工具提供了两个基本功能：

文件传输

类似Flume（在本章后面有详细描述）和DistCp（在本章后面有详细描述）这样的工具可以帮助用户将文件和纯文本（比如长条目）移动到Hadoop集群上。

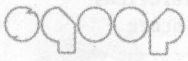
数据库传输

类似Sqoop（接下来将要进行详细描述）这样的工具为在传统的关系数据库（如Oracle 或 SQL Server）和Hadoop集群之间移动数据提供了一个简单的机制。

数据分类

类似 Storm（在本章后面有详细描述）这样的工具可以快速地对到达 Hadoop 系统上的新数据进行评价和分类。

6.1 Sqoop



许可证	Apache License, Version 2.0
活跃度	高
用途	在 HDFS 与关系数据库之间传递数据
官方网站	http://sqoop.apache.org
Hadoop 集成度	完全集成

你的数据很有可能起源于一个关系数据库管理系统(RDMBS)，该系统通常是通过 SQL 来访问的。同样你也可以使用 SQL 引擎来产生平面文件加载到 HDFS 上。尽管仓库加载大数据集更快，但你也有理由直接从 RDMBS 上获取数据或者将你的 Hadoop 上处理的结果放到 RDBMS 上。Sqoop（意思是 SQL to Hadoop）的设计就是用来在 Hadoop 和关系数据库之间进行数据传输的。它是 Cloudera 公司自行研制的 Apache 顶级项目，现在已经进入公共领域了。虽然 Sqoop 很多过程都是自动化的，但为了正常工作了解一些 SQL 知识仍然是必需的。接下来，将把 Sqoop 任务转化为一个 MapReduce 任务来工作。

你可以从向 Hadoop 上导入一个数据库表格开始，该表格作为一个文本文件或以 Avro 格式或以 SequenceFile 格式读入 Hadoop。你同样可以将一个 HDFS 文件导出到一个 RDBMS 中。在这种情况下，MapReduce 工作在 HDFS 上并行的读取一组以文本分隔的文件并将它们转化为 RDMBS 中的行。这里有一些选项来过滤行和列、修改分隔符等。

6.1.1 教程链接

在 YouTube 上可以获取一些非常好的关于这个话题的演讲。当你看过 Apache Sqoop 教程第 1 部分（*http://youtu.be/8NzcZzCrOcU*），就可以直接跳转到第 2、3、4 部分。

6.1.2 示例代码

在一个 PostgreSQL 数据库的表格上有我们的电影评论数据集，我们希望将它们导入到 Hadoop 上的文本文件中（同样我们也可以将这些数据从 Hadoop 上移动

到 RDBMS 上，但是不在这里介绍）：

```
myschema=> select * from moviereviews
reviewer  |     title      | score
----------+----------------+-------
Kevin     | Dune           |    10
Kevin     | Casablanca     |     5
Bob       | Blazing Saddles|     9
Marshall  | Dune           |     1


sqoop import --connect jdbc:postgresql://<host>/<database> \
    --table moviereviews --username JoeUser --P

<lots of lines omitted>

hadoop fs -cat moviereviews/part-m-00000
Kevin,Dune,10
Kevin,Casablanca,5
Bob, Blazing Saddles,9
Marshall,Dune,1
```

6.2 Flume

许可证	Apache License, Version 2.0
活跃度	中
用途	数据收集和聚合，尤其针对日志数据
官方网站	http:// ume.apache.org
Hadoop 集成度	完全集成

在数据收集系统中有一些确定的数据，你需要在Hadoop集群上对它们进行分析，现在需要做的就是找到一个将它们移到 Hadoop 集群上的方法。一般而言，你不能使用FTP或者SCP，因为它们只能在POSIX兼容的文件系统之间进行数据传输，但是 HDFS 并不是 POSIX 兼容的。当然，有一些 Hadoop 版本，比如 MapR 版本或者那些认证使用 IsilonOneFS 的版本，就可以兼容 POSIX。你可以 FTP 数据到 Hadoop 节点上的本地文件系统中，然后使用类似 copyFromLocal 这样的 HDFS 指令，但是这是冗长的，单线程的。Flume 解救了它们!

Flume 是一个可靠的分布式系统，它可以采集、聚合并且将大数据量的日志数据从多个来源上移动到 HDFS 上。它支持复杂的多次反射流以及扇入和扇出。在每一个 agent，event 都放在一个 channel 中，并通过该链交付给下一个 agent，channel 将一直保留 event 直到它们成功到达下一个 agent 或者 sink 的末端 HDFS 时才会被删除。一个 Flume 过程有一个配置文件，罗列了数据流的 source、 sink 以及 channel。最典型的用例包括将日志数据加载到 Hadoop 上。

6.2.1 教程链接

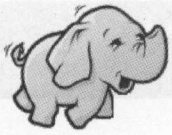
Dobb 博士的杂志发表了一篇关于 Flume 的报道性文章（*http://ubm.io/199uad3*）。如果读者对讲座感兴趣的话可以查看这个来自 2011 年的有趣的演讲（*http://youtu.be/POJCV28UYe4*）。

6.2.2 示例代码

使用 Flume，首先要构建用来描述 agent 的配置文件，包括 source、sink，以及 channel。这里 source 是 netcat，程序通过 TCP 来回显输出，sink 是一个 HDFS 文件，channel 是内存 channel：

```
# xmpl.conf

# 在这个 agent 中组件的名称
agent1.sources = src1
agent1.sinks = snk1
agent1.channels = chn1

# 描述 / 配置 source
agent1.sources.src1.type = exec
agent.sources.src1.command = tail -F /var/log/system.log
agent.sources.src1.channels = memory-channel

# 描述 sink
agent1.sinks.snk1.channel = memory-channel
agent1.sinks.snk1.type = hdfs
agent1.sinks.snk1.hdfs.path = hdfs://n1:54310/tmp/system.log/
agent1.sinks.snk1.hdfs.fileType = DataStream

# 使用一个 channel 在内存中缓冲事件
agent1.channels.chn1.type = memory
agent1.channels.chn1.capacity = 1000
agent1.channels.chn1.transactionCapacity = 100

# 将 source 和 sink 绑定到 channel 上
agent1.sources.src1.channels = c1
agent1.sinks.snk1.channel = c1

# 接下来开始 agent。将行添加到日志文件中，
# 它们将推送到内存 channel 中然后到 HDFS file_ 中

flume-ng agent --confconf --conf-file xmpl.conf --name agent1 \
    -Dflume.root.logger=INFO,console
```

6.3 DistCp



许可证	Apache License, Version 2.0
活跃度	低
用途	在 Hadoop 集群之间移动数据
官方网站	http://hadoop.apache.org/docs/r1.2.1/distcp2.html
Hadoop 集成度	完全集成

如果有一个 Hadoop 集群并且担心如果整个集群变得不可用将会发生什么事情，那么就有灾难恢复（DR）或操作连续性（COOP）问题。有几种策略可以处理这样的事情。一种解决方法是将所有的数据都同时加载到一个主要的 Hadoop 集群上和一个距离主集群遥远的备份集群上。这通常被称为双摄取。接下来你将不得不在主集群和远程集群上同时运行每一个任务以保持结果文件的一致。尽管该方法可行，但管理起来却十分复杂。你可能会想要去考虑使用 Apache Hadoop 中名为 DistCp 的内置部分。DistCP（Distributedcopy 的缩写）是在 Hadoop 集群之间移动数据的主要工具。同时，你可能还因为其他一些原因想要使用 DistCp，比如将数据从测试或开发集群上移到生产集群上。Hadoop 的商业版本都有处理 DR 和 COOP 的工具。很多都是构建在 DistCp 之上的。

6.3.1 教程链接

可能是由于 DistCp 较专一、简单的原因，没有很多有关该技术的教程。如果读者有兴趣更深一步的了解可以查看项目的官方网站（*http://bit.ly/1AmEdG0*）。

6.3.2 示例代码

这里要将源系统 `n1` 上的在 *source-dir* 目录中的 *source- file* 文件复制到 `n2` 上，其中 `n1` 和 `n2` 是源和目标的 NameNode 节点的主机名。如果你在一个 DR 情况下使用这些代码，*source-dir* 和 *dest-dir* 就是相同的，就好像 *source- file* 和 *dest- file* 这样：

```
$ hadoopdistcp hdfs://n1:8020/source-dir/source-file \
                    hdfs://n2:8020/dest-dir/dest-file
```

6.4 Storm

许可证	Apache License, Version 2.0
活跃度	高
用途	流摄取
官方网站	http://storm.apache.org
Hadoop 集成度	API 兼容

包括 Hadoop MapReduce 在内的很多大数据生态系统技术都被考虑用来去构建非常大的任务。这些系统被设计来执行批量的工作，将较小的任务捆绑到大的任务上，再来对这些大的任务进行分配。

尽管以分布式容错的方式来执行那些大数据量的复杂的分析，批处理是一个行之有效的策略，但它并不适合处理实时数据。这里就引入了类似 Storm 这样的系统。Storm 是流处理模型而不是批处理模型。这就意味着它的设计是用来对那些大数据量的小记录快速执行相对简单的转换。

在 Storm 上，一个工作流被称为“topology”，输入被称为“spouts”且转换被称为“bolts”。这里很重要的一点是要注意到 Storm 的 topology 与 MapReduce 的 job 是不同的，因为 job 有一个开始和结束，而 topology 是没有的。它的目的是一旦你定义了一个 topology，数据将会继续从你的 spout 流入且通过一些列的 bolts 进行处理。

6.4.1 教程链接

除了官方的 Storm 教程（*http://bit.ly/1FyPYt9*），在 GitHub 上的 Storm-Starter 项目（*http://bit.ly/1Ac5fl5*）上还有一系列非常好的入门资源。

6.4.2 示例代码

在这个例子中，我们要构建一个 topology，让它从一个 ReviewSpout 上来读取以逗号分隔的评论并且跟踪每一个标题被评论的次数。这里只涉及一点点有关

Storm topology 的定义，所以我们只介绍最重要的部分。

第一步就是通过定义一个 topology 来定义我们的输入。我们通过和我们的 topology 相关联一个 spout 来完成这件事。这个 spout 将负责从多源头读取数据，比如从一个 Twitter 或者 RSS 流入。

一旦我们定义了一个 spout，我们就可以开始定义 bolt 了。Bolt 负责对我们的数据进行处理。在这个例子中，我们有两个 bolt，第一个从评论中提取电影标题，而第二个计算每一个标题出现的次数：

```
TopologyBuilder builder = new TopologyBuilder();
builder.setSpout("review_spout", new ReviewSpout(), 10);
builder.setBolt("extract_title", new TitleBolt(), 8);
builder.setBolt("count", new TitleCount(), 15);

// 建立“conf”对象，并为你的工作适当地配置它

StormSubmitter.submitTopology("review_counter", conf,
builder.createTopology());
```

可以使用多种语言来编写 spout 和 bolt，甚至可以在一个 topology 中使用多种混合语言。举个例子，topology 是用 Java 语言来编写的，但是接下来要对一个 bolt 用 Python 语言来编写。这个 bolt 通过逗号来分隔评论并对第二字段进行检索，以此来从一个评论中提取电影的标题：

```
import storm

classTitleBolt(storm.BasicBolt):
    def process(self, tuple):
        words = tuple.values[0].split(",")
        storm.emit([words[1]])

TitleBolt().run()
```

第 7 章

安全、访问控制和审计

在 Hadoop 刚起步的时候，它的基础的安全模型被描述为，“在大象的周围修建栅栏，但是一旦进入到栅栏里面了，它的安全就有点松懈了。”尽管 HDFS 有访问控制机制，但是在 Hadoop 的世界里面，安全还是有点松懈了。近来，随着 Hadoop 变得越来越主流，安全问题正通过开发新工具来解决，比如 Sentry 和 Knox，以及已经确定了的类似 Kerberos 的机制。

根据 HIPAA、FiSMA 法案以及 PCI 的要求，大多数成熟的计算机系统都有访问、授权、加密及审计记录的方法。

身份验证回答了这样一个问题，“你是谁？”传统的强大的身份认证方法包括 Kerberos、轻量目录访问协议（Lightweight Directory Access Protocol，LDAP）以及活动目录（Active Directory，AD）。它们都是在 Hadoop 的外部来完成的，通常情况下是在客户端站点，如果适合的话也有可能是在 Web 服务器内。

授权回答了这样一个问题，“你能干什么？”在这里，Hadoop 遍布在各个地方。例如，MapReduce 工作采用队列系统存储其授权的方式与 HDFS 不同，HDFS 为用户 / 组 / 其他采用了常见的读 / 写 / 执行权限。HBase 有列族和表级授权，Accumulo 有单元级授权。

数据保护通常是指加密，无论是静态数据还是动态数据。HTTP、RPC、JDBC 以及 ODBC 都为数据传输及连接提供了加密。HDFS 目前没有本地加密，但是有一个提议在未来的发行版本中会包含这个。

管理和审计现在由 Hadoop 中特定的组件完成。在 HDFS 和 MapReduce 中都有一些基本的机制，而 Hive 元存储提供了日志服务，Oozie 提供了日志的作业管理服务。

指南（*http://bit.ly/1zLTzU9*）是一个了解更多 Hadoop 安全知识的非常好的入门之地。

如今，Hadoop 变得越来越主流，安全问题正通过开发新工具来解决，比如 Sentry（在本章后面有详细描述）、Kerberos（在本章后面有详细描述），以及 Knox（在本章后面有详细描述）。

7.1 Sentry

许可证	Apache License, Version 2.0
活跃度	高
用途	为 Hadoop 提供一个基础级授权
官方网站	https://incubator.apache.org/projects/sentry.html
Hadoop 集成度	API 兼容的孵化器项目（项目正在进行中）

如果在 Hadoop 上需要一个身份验证服务，那么 Sentry 就是其中一种可能的选择，它是 Apache 孵化器项目在 Hadoop 生态系统上提供的一个认证服务组件。目前，该系统在一个文件中定义了一些列的策略规则，包括定义组、映射组规则以及资源组特权规则。你可以看成是基于角色的访问控制（RBAC）。接下来应用程序将调用一个 Sentry API，包括用户的名字、该用户希望访问的资源以及访问方式。Sentry 的策略引擎将查看该用户是否属于这样一个群组，该群组中的角色有权限通过这样的请求方式来访问这些资源。它给应用程序返回一个二进制的是 / 否的结果，然后应用程序就可以根据该结果做出适当的反应。

目前，它是基于文件系统的且与 Hive 和 Impala 一起即插即用。其他的一些组件都可以使用 API。这个系统的一个缺点是，人们可以通过编写一个恶意的 MapReduce 程序来访问数据，这将限制 Hive 接口数据的使用。

孵化器项目不是官网发布的 Hadoop 中的一部分，它不应该使用在生产系统中。

7.1.1 教程链接

在 Apache 的官方微博中有两个非常好的帖子。第一个帖子（*http://bit.ly/1EYsznm*）提供了这项技术的概述，而第二个帖子（*http://bit.ly/1KJe9rE*）是一个入门指南。

7.1.2 示例代码

Sentry 的配置过程相当复杂，已经超出了我们这本书的范围。Apache 博客提供了一个非常好的资源，读者可以通过阅读该资源开始使用这项技术。

在这个 Apache 博客教程（*http://bit.ly/1KJe9rE*）中有非常简洁的示例代码。

7.2 Kerberos

许可证	MIT license
活跃度	高
用途	安全认证
官方网站	http://web.mit.edu/kerberos
Hadoop 集成度	API 兼容

在 Hadoop 集群上进行身份认证的一个常见的方法是，使用一个名为 Kerberos 的安全工具。Kerberos 是一个基于网络的工具，是由麻省理工学院发布的用来提供在客户端请求访问和服务器提供访问之间的基于提供安全加密票据的强认证。

这个模型相当简单。客户在 Kerberos 的密钥分发中心（KDC）注册并共享它的密码。当一个客户想要访问一个类似文件服务器的资源时，它将向 KDC 发送请求且该请求的一部分用此密码进行加密。KDC 将尝试对其进行解密，如果成功了，则向客户发送回一个票据生成的票据（TGT），它将通过它特殊的密码进行加密。当客户端收到 TGT，它将向 KDC 发送回一个访问文件服务器的请求。KDC 发送回一张通过文件服务器的密码进行位加密的票据。在这之后，客户端和文件服务器将使用这个票据进行身份验证。

文件服务器可能会因过多的客户请求变得非常繁忙，有个概念是文件服务器不会因为过多的用户密码影响而变得停滞不前。

它只是与 KDC 分享它的密码，以及使用客户端从 KDC 接收到的票据来进行身份验证。

人们认为 Kerberos 的建设和维护是烦琐的。为此，在 Hadoop 社区里有很多活跃的工作为的是呈现一种更简单、更有效的身份验证机制。

7.2.1 教程链接

在 *http://youtu.be/kp5d8Yv3-0c* 上提供了一个演讲，为此技术给出了一个相当简洁且浅显易懂的描述。

7.2.2 示例代码

一个有效的Kerberos安装是一项非常艰巨的任务，且已经超过了本书的讨论范围。许多操作系统供应商都提供了一个有关 Kerberos 的配置指南。如需要获取等多的信息，可以参考特定的操作系统的指南。

7.3 Knox

许可证	Apache License, Version 2.0
活跃度	中
用途	安全网关
官方网站	https://knox.apache.org
Hadoop 集成度	完全集成

保护一个 Hadoop 集群往往是非常复杂的、耗费时间的，在充满了权衡和妥协中努力。在这个挑战中的最大的因素是 Hadoop 中有非常多的不同的技术，而每一种都有它自己的安全概念。

保护一个集群的常见的方法是用防火墙（“大象的栅栏”）来简单地保护它的环境。这种方法在早期还是可以接受的，那时 Hadoop 很大程度上仅仅是数据科学家和信息分析师们使用的一个独立的工具，但是如今的 Hadoop 已经是大数据生态系统中的一部分，它以各种各样的方式与多种工具通过接口连接。不幸的是，每一个工具似乎都有它自己的公开接口，且如果提出一个安全模型，它往往与其他的任何工具皆不相同。这一切的最终结果是，使那些想要维持一个安全环境的用户发现自己想要通过防火墙来找到漏洞，以及试图管理各种各样不同的用户列表和工具配置都注定是一场失败的战争。

Knox 的设计目的就是为了应对这样的复杂性。它是一个独立的网关，位于外部系统与你的 Hadoop 集群之间以及集群内部。它还提供了一个独立的安全接口，拥有授权、身份验证和审计（AAA）能力，且与许多标准系统都有接口，如 ActiveDirectory 和 LDAP。

7.3.1 教程链接

Hortonworks 公司的员工为获取一个最小的 Knox 网关提供了一个非常简明的指南（*http://bit.ly/1uEYW76*）。如果你有兴趣继续深入学习，在 Knox 官网（*https://knox.apache.org*）上的官方的快速入门指南提供了大量的细节。

7.3.2 示例代码

即使是 Knox 的一个简单的配置也超过了本书讨论的范围。这里鼓励有兴趣的读者可以查看教程和快速入门。

第8章

云计算和虚拟化

当前大多数Hadoop集群都运行在“裸金属”架构上，这是指这些集群运行在一些基于Intel的计算机上，这些计算机使用各种各样的Linux操作系统，并配了直接连接的存储（directly attached storage，DAS）。然而，你可能希望尝试将Hadoop集群运行在云计算或虚拟化环境中。尽管虚拟化技术通常在性能上会有某种程度的下降，但这对于你的任务来说影响可能是很小的，或者说这种开销对于获取云计算带来的收益来说是值得的。这些收益包括降低预付开销，以及可以随着数据集和分析需求的改变而扩展（有时候是收缩）的能力。

对于云计算，本书遵循美国国家标准技术研究所（National Institute of Standards and Technology，NIST）的规范，NIST给出了云计算的定义，详见“*http://bit.ly/nist-cloud-compute*”。一个在云中的Hadoop集群通常具有如下特性：

- 按需获取自助服务。
- 网络访问。
- 资源共享。
- 快速伸缩。
- 可度量资源服务。

尽管这些资源需求不是真正的存在，在实践中，他们通常是这么做的。

虚拟化意味着创建虚拟的计算机实体，与之相对应的是真实的计算机实体。通常来说，虚拟的对象是一个安装了软件和应用程序的操作系统，但其实存储和网络

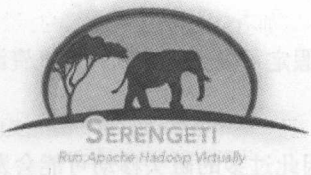
也可以是虚拟化的。也许你认为虚拟化是一门相对新的计算机技术，但实际上在1972年IBM公司就发布了VM/370，370大型机就可以分割为多个小的单用户虚拟机。现在，亚马逊的AWS服务可能是最知名的云计算基础设施。维基百科中有关于虚拟化的简要解释，网址是“*http://bit.ly/1Chsx4b*”。

Hadoop官方也提供了从Hadoop视角对云计算和虚拟化的解释，详见维基百科对应页面，网址是“*http://bit.ly/1yhiNoe*”。一个关于Hadoop的指导原则是数据分析应该运行在集群中靠近数据的节点上。为什么？因为在集群中传输数据块会降低性能。HDFS中每个文件块通常会保存3份，MapReduce很可能选择数据存储的数据节点来运行任务。在一个简单虚拟环境中，数据的物理位置是未知的，实际上，真实的物理存储可能在某个位置，该位置不位于集群中任何一个节点上。

可以查看*http://vmw.re/1KJesCR*来获取关于虚拟化Hadoop很好的背景知识，尽管这是从VMware公司角度提出来的。

在本章，你将看到一些开源软件，这些软件促进了云计算和虚拟化。也有其他专用的解决方案，但是在本书中并不涉及。

8.1 Serengeti



许可证	Apache License，Version 2.0
活跃度	中
用途	Hadoop 虚拟化
官方网站	http://www.projectserengeti.org
Hadoop 集成度	未集成

如果你的组织使用 VMware 公司的 vSphere 作为虚拟化方案的基础，那么 Serengeti 为你提供了一种在现有环境中快速构建 Hadoop 集群的方法。众所周知，vSphere 是一个专利保护的环境，但是在该环境中运行 Hadoop 的代码是开源的。尽管 Serengeti 并不附属于 Apache 软件基金会（Apache 软件基金会还管理了许多其他和 Hadoop 相关的项目），但已经有许多人成功地将 Serengeti 用于快速部署了。

为什么要虚拟化 Hadoop 呢？从历史的角度看，Hadoop 集群运行在普通服务器上（也就是 Intel x86 服务器，服务器配有自己的硬盘，运行 Linux 操作系统）。当编制计划任务时，Hadoop 利用 HDFS（详见第 1 章描述）中数据的位置，来使得代码尽可能靠近数据，倾向于代码运行的服务器和数据所在服务器是同一台，这样就能最小化通过网络传输的数据量。在许多虚拟化环境中，直接附加存储被通用存储设备所代替，通常是存储区域网络（storage area network，SAN）或网络附加存储（network attached storage，NAS）。在这些环境中，没有存储位置的概念。

有很多使用虚拟化 Hadoop 的原因，优点如下，现在似乎已经有许多 Hadoop 集群运行在公有云之上了：

- 可以加快配置集群的速度，用户不需要购买和配置硬件。
- 可以快速增加和快速减少集群的大小，以应对服务的需求。

- 通过使用虚拟化技术进行管理，可以提高容错性，快速从错误中恢复。

同时也有一些缺点：

- MapReduce 和 YARN 假定可以完全控制计算机资源。而在虚拟化环境中并不是如此。
- 数据布局十分关键，因此过多的磁头移动可能会发生，而标准的三副本镜像对于数据保护十分重要。一个好的虚拟化策略必须有一致性。有些做，有些不做。

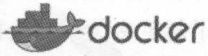
你需要权衡优势和缺点，来决定你的项目是否适合虚拟化的 Hadoop。

8.1.1 教程链接

可以从如下链接中获取关于虚拟化 Hadoop 的背景知识：

- “使用 Serengeti 配置 Hadoop”（*http://bit.ly/1CVnHPs*）。
- “Vitual Hadoop 的维基百科页面”（*http://bit.ly/1yhiNoe*）。
- “在 VMware vSphere 5 上的 Hadoop 虚拟化扩展”（*http://vmw.re/1tOgJbj*）。
- “虚拟化 Apache Hadoop”（*http://vmw.re/1KJesCR*）。

8.2 Docker



许可证	Apache License，Version 2.0
活跃度	高
用途	运行应用程序的容器，也包括 Hadoop 节点
官方网站	https://www.docker.com
Hadoop 集成度	未集成

你应该听说过口口相传的 Docker 和容器化的应用程序。这里介绍一小段历史，可能可以帮助你更好地理解 Docker。不论是对于云计算，还是对于基础设施即服务（infrastructure as a service，IaaS），虚拟机都是非常重要的一步。一旦创建一个 Linux 虚拟机，就可以很方便的根据这个虚拟机创建一个新的虚拟机，然而要构建一个 Linux 硬件机器可能需要几个小时的时间。但是虚拟化也有一些缺点，如果已经构建了 100 个虚拟机的集群，这时需要修改操作系统的一个参数，或者需要对 Hadoop 环境中进行某些更新操作，那么将需要对这所有 100 个虚拟机进行逐个修改。

为了弄清楚 Docker 的优势所在，理解一下它的来龙去脉是很有用的。首先出现的是 chroot jails，在这个 Unix 子系统中，可以构建程序并将其限制在一个较小的名称空间中，而不是整个 Unix 操作系统。然后出现了 Solaris 容器，通过使用该容器，用户和程序被限制在某个区域内，使用虚拟资源来进行保护。Linux 容器大体上是和 Solaris 容器一样，区别仅仅在于使用的是 Linux 系统而不是 Solaris。Docker 是作为一个应用程序轻量级虚拟化技术出现的。Docker 容器位于 Linux 操作系统资源最顶层，并管理应用程序代码及其依赖的任何操作系统资源。因此 Docker 可以享受虚拟机的资源独立性和资源分配特性，但移植性更好，更轻量级。对于 Docker 的完整描述已经超出了本书的范围，但是最近已经有人尝试将 Hadoop 节点运行在 Docker 环境中。

Docker 是一门很新的技术。目前还不清楚该技术是否已经为大规模的 Hadoop 生产环节做好准备。

8.2.1 教程链接

Docker 使用人员已经提供了一个交互式教程（*http://bit.ly/1DFq08i*），这将使得工作更为简单。读者如果希望了解更多关于 Docker 背后的容器技术，可以查看 DeveloperWork 文档（*http://ibm.co/1E9F5MY*），该文档十分有趣。

8.2.2 示例代码

之前提供的指南为提供运行 Docker 范例做了很好的工作。Pivotal 的博客（*http://bit.ly/1EYt4hr*）展示了在 Docker 上配置 Hadoop 的一个例子。

8.3 Whirr

许可证	Apache License，Version 2.0
活跃度	低
用途	配置集群
官方网站	https://whirr.apache.org
Hadoop 集成度	API 兼容

构建一个大数据集群是十分昂贵的、耗时的、复杂的事情，通常需要多个团队协调完成。有时你仅仅需要加速创建一个集群来测试某种能力或某个想法的原型。类似 Amazon 的 EC2 或 Rackspace 云服务都提供了一种实现该功能的方法。不幸的是，与不同的提供商相连的接口差异性很大。因此一旦你围绕着构建和关闭测试集群的进程开发了某些自动化程序，你实际上已经将自己绑定在某个服务提供商了。Apache Whirr 为几个不同的服务提供商进行交互提供了一个标准的机制。这使得你可以简单地修改云服务提供商，或者和其他不使用相同云服务提供商的团队共享配置。

Whirr 的最基本的构件块就是实例模板。*Instance templates* 定义了一个用途。举例来说，现在有许多个模板，为 Hadoop 的 jobtracker、ZooKeeper 和 Hbase region 节点。*Recipes* 是一个选取模板并定义集群的步骤。举例来说，要创建一个简单的数据处理集群，可以使用一个 *recipe*，来配置 Hadoop NameNode、Hadoop jobtracker、一对 ZooKeeper 服务器、一个 HBase master 和几个 HBase region 服务器。

8.3.1 教程链接

Apache Whirr 的官方网站（*https://whirr.apache.org/*）提供了两个非常优秀的指南。“Whirr in 5 Minutes”（*http://bit.ly/17gPvAF*）指南提供了创建和关闭第一个集群所需要的准确的命令。“quick-start guide”（*http://bit.ly/Whirr_quick-start*）

稍微有些复杂，贯穿说明了处理中每个阶段发生的事情。

8.3.2 示例代码

既然这样，我们一起来配置一个之前描述过的简单的数据集群，使用一个已经建立的 Amazon EC2 账户。

第一步是构建文明的 recipe 文件（我们将该文件命名为 *field_guide.properties*）：

```
# field_guide.properties

# 这是给集群起的名字，用于和其他云服务提供商进行通信
whirr.cluster-name=field_guide

# 因为这里仅仅是进行测试，所以我们将所有 master 放在单个机器上，
# 并仅构建 3 个 worker 节点
whirr.instance-templates= \
1 zookeeper+hadoop-namenode \
+hadoop-jobtracker \
+hbase-master,\
3 hadoop-datanode \
+hadoop-tasktracker \
+hbase-regionserver

# 现在在 Amazon EC2 上配置该集群
whirr.provider=aws-ec2

# 根据所选择的提供商不同，身份和证书信息都会有所不同
# 因为这里我们使用 EC2，我们需要输入我们的 Amazon 访问 ID 和密码
# 这可以很容易的从提供商获取
whirr.identity=<your identity here>
whirr.credential=<your key here>

# 这是我们要勇于访问集群的证书
# 在该情形，Whirr 将在集群中每台机器上创建一个名为 field_guide_user 的用户，
# 使用我们的 ssh 公钥来连接用户
whirr.cluster-user=field_guide_user
whirr.private-key-file=${sys:user.home}/.ssh/id_rsa
whirr.public-key-file=${sys:user.home}/.ssh/id_rsa
```

```
# 现在已经做好配置集群的准备了
# 为了做到这点，我们简单地运行 whirr，使用“launch cluster”参数，
# 并把我们的 recipe 传递给它
$ whirr launch-cluster --config field_guide.properties

# 一旦集群处理完毕，用户可能希望关闭该集群
# 我们运行一个类似的命令，这次传递一个名为“destroy-cluster ”的参数
$ whirr destroy-cluster --config field_guide.properties
```

作者介绍

Kevin Sitto是Pivotal Software公司的领域解决方案工程师，他为用户提供咨询服务，帮助用户理解和描述他们的大数据需求。

他和妻子以及两个孩子住在Maryland，在没有撰写关于大数据书籍的时候，他经常享受制作自酿的啤酒的乐趣。

Marshall Presser是Pivotal Software公司的领域首席技术官，住在弗吉尼杰州麦克莱恩市。除了帮助用户使用Greenplum数据库解决复杂的分析问题之外，他领导了Hadoop Vitual 领域团队，工作关注于将Hadoop与关系型数据库整合。

在来到Pivotal公司（之前是Greenplum公司）之前，他在Oracle工作了12年，专门从事于高可用性、业务连续性、集群、并行数据库技术、灾难恢复和大规模数据库系统。Marshall之前还为许多硬件厂商实现集群和其他并行体系架构。他的背景包括并行计算和操作系统/编译器开发，同时还是健康医疗机构、金融服务机构、联邦政府和州政府的私人顾问。

Marshall获得了Pennsylvania大学的数学学士学位和金融与统计学硕士学位，同时还获取了伦敦Imperial大学的计算机理科硕士。

封面介绍

本书的封面的动物是O'Reilly动物，大多数是与本书涉及的技术相关联的，包括：贼鸥海鸟（skua seabird）、沼泽无尾刺豚鼠（lowland paca）、九头蛇波西亚帕西菲卡（hydra portia pacific）、炮弹鱼（trigger fish）、非洲大象（African elephant）、麋鹿（Pere David's deer）、欧洲野猫（European wildcat）、披肩鸡（ruffed grouse）和黑猩猩（chimpanzee）。

O'Reilly封面的大多数动物都濒临灭绝，对于地球来说它们都是很重要的。如果希望了解更多如何帮助它们，可以访问animals.oreilly.com。

O'REILLY®

Hadoop生态系统

如果你的组织即将开始进入大数据的世界，那么可能不仅需要决定Apache Hadoop这个平台是否适合使用，还需要决定Hadoop中哪些组件最适合完成你的任务。本书将帮助你更容易地完成这项工作。本书将Hadoop的生态系统分解为一个个简略的、容易理解的小段内容，以便读者可以快速理解Hadoop项目、子项目及其相关技术是如何一起工作的。

本书每一章都介绍了不同的主题（例如核心技术或数据传输），并且解释了为什么特定组件适用或不适用特定的需求。对于数据处理来说，使用Hadoop是一个全新的挑战，但如果有了这本便利的参考书，你将很容易领会使用Hadoop的精妙所在。

主要包括如下主题：

- **核心技术**。Hadoop分布式文件系统（HDFS）、MapReduce、YARN和Spark。
- **数据库和数据管理**。Cassandra、HBase、MongoDB和Hive。
- **序列化**。Avro、JSON和Parquet。
- **管理和监视**。Puppet、Chef、Zookeeper和Oozie。
- **分析辅助**。Pig、Mahout和MLLib。
- **数据传输**。Scoop、Flume、distcp和Storm。
- **安全、访问控制和审计**。Sentry、Kerberos和Knox。
- **云计算和虚拟化**。Serengeti、Docker和Whirr。

Kevin Sitto 是Pivotal Software公司的领域解决方案工程师，主要为客户提供咨询服务，帮助客户理解和描述大数据需求。

Marshall Presser 是Pivotal Data Engineering集团的成员。他帮助客户使用Hadoop、关系数据库和内存数据网格来解决复杂的分析问题。

DATA | HADOOP

O'Reilly Media, Inc.授权中国电力出版社出版

此简体中文版仅限于在中华人民共和国境内（但不允许在中国香港、澳门特别行政区和中国台湾地区）销售发行

This Authorized Edition for sale only in the territory of People's Republic of China (excluding Hong Kong, Macao and Taiwan)

ISBN 978-7-5123-9598-5

9 787512 395985

定价：28.00元